Similarity Search
The pq-Gram Distance

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

Unit 11 – May 24, 2012

Outline

The pq-Gram Distance
- Definition
- Algorithm
- Fanout Weighting and Lower Bound
- Experiments

pq-Grams – Intuition

- q-Grams for strings:
  - split string into substrings (q-grams) of length q
  - strings with many common substrings are similar
- pq-Grams for trees:
  - split tree into small subtrees (pq-grams) of the same shape
  - trees with many common subtrees are similar
**pq-Grams**

- The shape of a **pq-gram** ($p = 2, q = 3$):
  - $p$ nodes (anchor node and $p - 1$ ancestors) form the **stem**
  - $q$ nodes (consecutive children of the anchor node) form the **base**

**pq-Extended Tree**

- **Problem:** How can we split the following tree $T$ into 2,3-grams?

- **Solution:** Extend tree $T$ with dummy nodes (●):
  - $p - 1$ ancestors to the root node
  - $q - 1$ children before the first and after the last child of each non-leaf
  - $q$ children for each leaf

- The result is the **pq-extended tree** $T_{pq}$.

**Example: Extended Tree**

An example tree $T$ and its extended tree $T_{pq}$ ($p = 2, q = 3$):

**Definition: pq-Gram** [ABG05]

Let $T$ be a tree, $T_{pq}$ the respective extended tree, $p > 0, q > 0$. A subtree of $T_{pq}$ is a **pq-gram** $g$ of $T$ iff:

- (a) $g$ has $q$ leaf nodes and $p$ non-leaf nodes,
- (b) all leaf nodes of $g$ are children of a single node $a \in N(g)$ with fanout $q$, called the **anchor node**,
- (c) the leaf nodes of $g$ are consecutive siblings in $T_{pq}$.

- **Stem:** anchor node and its ancestors in the **pq-gram**.
- **Base:** children of the anchor node in the **pq-gram**.

**Definition (pq-Gram Profile)**

The **pq-gram profile**, $P_T$, of a tree $T$ is the set of all its **pq-grams**.
**pq-Gram Distance**

**Definition**

Example: Systematically Split Tree

- **pq-Gram**: small subtree with stem and base
  - Example: $p = 2$, $q = 3$
  - Systematically split tree into pq-grams
- **pq-Gram profile**: set of all pq-grams of a tree.

**Label Tuples**

- **Linear encoding** of a pq-gram $g$ with anchor node $v_p$:
  - (traverse pq-gram in preorder)
  - $v_1, \ldots, v_p, v_{p+1}, \ldots, v_{p+q}$
- **Label tuple**: tuple of the pq-gram’s node labels
  - $\lambda(g) = (\lambda(v_1), \ldots, \lambda(v_{p+q}))$
  - for the pq-gram $g = (v_1, \ldots, v_{p+q})$.

**Storing the pq-Gram Index Efficiently**

**Problem**: How to store node labels efficiently?

- **Long labels**: large storage overhead
- **Varying label length**: in a relational database, the inefficient VARCHAR type must be used instead of the efficient CHAR type

**Solution**: Hashing

- compute fingerprint hash for labels
- store concatenation of the hashed labels

**Fingerprint hash function** (e.g., Karp-Rabin [KR87]):

- maps a string $s$ to a hash value $h(s)$
- $h(s)$ is of fixed length
- $h(s)$ is unique with high probability
  - (for two different strings $s_1 \neq s_2$, $h(s_1) \neq h(s_2)$ with high probability)
### Definition (pq-Gram Distance)

The pq-gram distance between two trees, \( T \) and \( T' \), is defined as

\[
\delta_g(T, T') = |I(T) \cup I(T')| - 2|I(T) \cap I(T')|
\]

- **Metric normalization** to \([0..1]\): \( \delta_g'(T, T') = \frac{\delta_g(T, T')}{|I(T) \cup I(T')| - |I(T) \cap I(T')|} \)
- **Pseudo-metric** properties hold for normalization [ABG10]:
  - **self-identity**: \( x = y \not\Rightarrow \delta_g(x, y) = 0 \)
  - **symmetry**: \( \delta_g(x, y) = \delta_g(y, x) \)
  - **triangle inequality**: \( \delta_g(x, z) \leq \delta_g(x, y) + \delta_g(y, z) \)
- Different trees may have identical indexes:

\[
\begin{align*}
\{a, a\ast a, a\ast ab, a\ast abc, *ac\ast*, aa\ast e, aa\ast eb, \\
aaeb\ast, aab\ast b, ab\ast, ac\ast, ae\ast, ab\ast\ast\}
\end{align*}
\]

Intuition: similar trees have similar pq-gram indexes.

### Main Memory Algorithm (I)

**CREATE_INDEX**(\( T, r, I, \text{stem}, p, q \))

\[
\begin{align*}
\text{stem} & := \text{shift}(\text{stem}, \lambda(r)) \\
\text{base} & := \text{shift register of size } q \text{ (filled with *}) \\
\text{if } r \text{ is a leaf} \text{ then} \\
I & := I \cup \{\text{stem} \cup \text{base}\} \\
\text{else} \\
\text{for each} \ c \ (\text{from left to right}) \text{ of } r \text{ do} \\
\text{base} & := \text{shift (base, } \lambda(c)) \\
I & := I \cup \{\text{stem} \cup \text{base}\} \\
I & := \text{CREATE_INDEX}(T, c, I, \text{stem}, p, q) \\
\text{for } k := 1 \text{ to } q - 1 \\
\text{base} & := \text{shift (base, *}) \\
I & := I \cup \{\text{stem} \cup \text{base}\} \\
\text{return } I
\end{align*}
\]
The pq-Gram Distance Algorithm

Main Memory Algorithm (II)

Input of \texttt{createIndex}(T, r, I, stem, p, q):
- a subtree of T rooted in r
- the pq-gram index I computed so far
- the stem stem of r’s parent
- the parameters p and q

Output of \texttt{createIndex}(T, r, I, stem, p, q):
- the pq-gram index including the input index I
- the pq-gram index of r and all its descendants i.e., the pq-grams (label tuples) with anchor node r or a descendant of r

Nikolaus Augsten (DIS) Similarity Search Unit 11 – May 24, 2012 17 / 32

pq-Gram-Index

\texttt{pq-Gram-Index}(T, p, q) computes the pq-gram index for a complete tree T:

\begin{itemize}
  \item \texttt{stem}: shift register of size p (filled with *)
  \item \texttt{I}: empty index
  \item \texttt{I} = \texttt{createIndex}(T, root(T), I, stem, p, q)
  \item \texttt{return} I
\end{itemize}

Nikolaus Augsten (DIS) Similarity Search Unit 11 – May 24, 2012 18 / 32

Complexity of the pq-Gram Index Algorithm

Theorem (pq-Gram Index Complexity)
The pq-gram index of a tree T with size \(|T|\) can be computed in \(O(|T|)\) time.

Proof.
Each recursive call of \texttt{createIndex()} processes one node in constant time, and each node is processed exactly once.

Nikolaus Augsten (DIS) Similarity Search Unit 11 – May 24, 2012 19 / 32

Size of the pq-Gram Index

Theorem (Size of the pq-Gram Index)

Let T be a tree with l leaves and i non-leaves. The size of the pq-gram index of T is

\[ |I_{pq}(T)| = 2l + qi - 1. \]

Proof.

1. We count all pq-grams whose leftmost leaf is a dummy node: Each leaf is the anchor node of exactly one pq-gram whose leftmost leaf is a dummy node, giving l pq-grams. Each non-leaf is the ancestor of \(q - 1\) pq-grams whose leftmost leaf is a dummy node, giving \(l(q - 1)\) pq-grams.
2. We count all pq-grams whose leftmost leaf is not a dummy node: Each node of the tree except the root is the leftmost leaf of exactly one pq-gram, giving \(i + q - 1\) pq-grams.

Overall number of pq-grams: \(l + i(q - 1) + (l + i - 1) = 2l + qi - 1.\)
Outline

1. The pq-Gram Distance
   - Definition
   - Algorithm
   - Fanout Weighting and Lower Bound
   - Experiments

Motivation: Unit Cost Model Not Always Intuitive

Unit cost edit distance:
- no difference between leaves and non-leaves
- may lead to non-intuitive results

Conclusion: Non-leafs should have more weight than leafs.

Fanout Weighted Tree Edit Distance

Definition (Fanout Weighted Tree Edit Distance)

Let T and T’ be two trees, w ∈ N(T) a node with fanout f, w’ ∈ N(T’) a node with fanout f’, c > 0 a constant. The fanout weighted tree edit distance, δ_f = (T, T’), between T and T’ is defined as the tree edit distance with the following costs for the edit operations:

- **Delete:** \( \alpha(w \to e) = f + c \)
- **Insert:** \( \alpha(e \to w') = f' + c \)
- **Rename:** \( \alpha(w \to w') = (f + f')/2 + c \)

- **Cost of changing a non-leaf node:** proportional to its fanout.
- **Cost of changing a leaf node:** constant c.

Example: Fanout-Weighted Tree Edit Distance

- Fanout-Weighted Tree Edit Distance:
  - leaf changes have small cost (c = 1 in the example)
  - non-leaf changes cost proportional to the node fanout
The pq-Gram Distance Lower Bound

Theorem

Let \( p = 1 \) and \( c \geq \max(2q - 1, 2) \) be the cost of changing a leaf node. The pq-gram distance provides a lower bound for the fanout weighted tree edit distance, i.e., for any two trees, \( T \) and \( T' \),

\[
\frac{\delta_g(T, T')}{2} \leq \delta_f(T, T').
\]

Proof.

See [ABG10] (ACM Transactions on Database Systems).

Size of the pq-Gram Index

- pq-Gram index size: linear in the tree size
- Experiment:
  - compute pq-gram index for trees with different number of nodes
  - compare tree and index size

Why is the pq-gram index smaller than the tree?
- hash values are smaller than labels
- duplicate pq-grams of a tree are stored only once

(Similarity Search Unit 11 – May 24, 2012 26 / 32)
Sensitivity to Structure Change — Non-Leaf

- Cost for non-leaf change → controlled by \( p \)
- **Experiment:**
  - delete non-leaf nodes
  - measure normalized \( pq \)-gram distance

Scalability to Large Trees

- \( pq \)-gram distance → scalable to large trees
- compare with edit distance
- **Experiment:** For pair of trees
  - compute tree edit distance and \( pq \)-gram distance
  - vary tree size: up \( 5 \times 10^5 \) nodes
  - measure wall clock time

**pq-Grams vs. other Edit Distance Approximations**

<table>
<thead>
<tr>
<th>Distance</th>
<th>Correct</th>
<th>Recall</th>
<th>Precision</th>
<th>f-Measure</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>fanout edit dist</td>
<td>259</td>
<td>86.6%</td>
<td>98.5%</td>
<td>0.922</td>
<td>19 min</td>
</tr>
<tr>
<td>unit edit dist</td>
<td>247</td>
<td>82.6%</td>
<td>96.5%</td>
<td>0.890</td>
<td>14 min</td>
</tr>
<tr>
<td>node intersection</td>
<td>197</td>
<td>65.9%</td>
<td>93.8%</td>
<td>0.774</td>
<td>4.3s</td>
</tr>
<tr>
<td>( p,q )-grams</td>
<td>236</td>
<td>78.9%</td>
<td>98.7%</td>
<td>0.877</td>
<td>8.1s</td>
</tr>
<tr>
<td>tree-embedding</td>
<td>206</td>
<td>68.9%</td>
<td>96.3%</td>
<td>0.803</td>
<td>7.1s</td>
</tr>
<tr>
<td>binary branch</td>
<td>193</td>
<td>64.5%</td>
<td>93.2%</td>
<td>0.763</td>
<td>7.4s</td>
</tr>
<tr>
<td>bottom-up</td>
<td>148</td>
<td>49.6%</td>
<td>92.5%</td>
<td>0.645</td>
<td>67.0s</td>
</tr>
</tbody>
</table>

Effectiveness: \( pq \)-grams outperform all other approximations
