Search Space Reduction for the Tree Edit Distance

Similarity Join and Search Space Reduction

Lower Bound: Traversal Strings

Upper Bound: Constrained Edit Distance

Definition: Similarity Join

Given two sets of trees, $S_1$ and $S_2$, and a distance threshold $\tau$, let $\delta_t(T_i, T_j)$ be a function that assesses the edit distance between two trees $T_i \in S_1$ and $T_j \in S_2$. The similarity join operation between two sets of trees reports in the output all pairs of trees $(T_i, T_j) \in S_1 \times S_2$ such that $\delta_t(T_i, T_j) \leq \tau$. 
SimJoin($S_1, S_2$)

for each $T_i \in S_1$ do
    for each $T_j \in S_2$ do
        if upperBound($T_i, T_j$) $\leq \tau$ then
            output($T_i, T_j$)
        else if lowerBound($T_i, T_j$) $> \tau$ then
            /* do nothing */
        else if $\delta_t(T_i, T_j)$ $\leq \tau$ then
            output($T_i, T_j$)

Outline

1. Search Space Reduction for the Tree Edit Distance
   - Similarity Join and Search Space Reduction
   - Lower Bound: Traversal Strings
   - Upper Bound: Constrained Edit Distance

Notes: Traversal Strings and Tree Inequality

- If the traversal strings of two trees are equal, the trees can still be different:
  
  $$T_1 = \begin{array}{c} \text{a} \\ \text{b} \end{array}$$  
  $$T_2 = \begin{array}{c} \text{a} \\ \text{b} \end{array}$$

  $$\text{pre}(T_1) = \text{aba} = \text{pre}(T_2) = \text{aba}$$
**Lower Bound**

**Theorem (Lower Bound)**

If the trees are at tree edit distance \( k \), then the string edit distance between their preorder or postorder traversals is at most \( k \).

**Proof.**

Tree operations map to string operations (illustration on next slide):

- **Insertion** \((\text{ins}(v, p, k, m))\): Let \( t_1 \ldots t_r \) be the subtrees rooted in the children of \( p \). Then the preorder traversal of the subtree rooted in \( p \) is \( p \text{pre}(t_1) \ldots \text{pre}(t_{k-1}) \text{pre}(t_k) \ldots \text{pre}(t_m) \text{pre}(t_{m+1}) \ldots \text{pre}(t_r) \).

  Inserting \( v \) moves the subtrees \( k \) to \( m \):

  \[
  p \text{pre}(t_1) \ldots \text{pre}(t_{k-1}) v \text{pre}(t_k) \ldots \text{pre}(t_m) \text{pre}(t_{m+1}) \ldots \text{pre}(t_r) .
  \]

  The string distance is 1. Analog rationale for postorder.

- **Deletion**: Inverse of insertion.

- **Rename**: With node rename a single string character is renamed.

From the lower bound theorem it follows that

\[
\max(\delta_s(\text{pre}(T_1), \text{pre}(T_2)), \delta_s(\text{post}(T_1), \text{post}(T_2))) \leq \delta_t(T_1, T_2)
\]

where \( \delta_s \) and \( \delta_t \) are the string and the tree edit distance, respectively.

The string edit distance can be computed faster:

- string edit distance runtime: \( O(n^2) \)
- tree edit distance runtime: \( O(n^4) \)

**Similarity join**: match all trees with \( \delta_t(T_1, T_2) \leq \tau \)

- if \( \max(\delta_s(\text{pre}(T_1), \text{pre}(T_2)), \delta_s(\text{post}(T_1), \text{post}(T_2))) > \tau \)

  then \( \delta_t(T_1, T_2) > \tau \)

  thus we do not have to compute the expensive tree edit distance.
The string distances of preorder and postorder may be different. The string distances and the tree distance may be different.

\[
\begin{align*}
T_1 & = \text{a} \text{b} \text{c} \\
T_2 & = \text{a} \text{b} \text{c}
\end{align*}
\]

\[
\begin{align*}
\text{pre}(T_1) & = \text{abac} \\
\text{pre}(T_2) & = \text{abac} \\
\text{post}(T_1) & = \text{bcaa} \\
\text{post}(T_2) & = \text{acba}
\end{align*}
\]

\[
\begin{align*}
\delta_s(\text{pre}(T_1), \text{pre}(T_2)) & = 0 \\
\delta_s(\text{post}(T_1), \text{post}(T_2)) & = 2 \\
\delta_t(T_1, T_2) & = 3
\end{align*}
\]

Definition (Edit Mapping)

An edit mapping \( M \) between \( T_1 \) and \( T_2 \) is a set of node pairs that satisfy the following conditions:

1. \((a, b) \in M \Rightarrow a \in N(T_1), b \in N(T_2)\)
2. For any two pairs \((a, b)\) and \((x, y)\) of \( M \):
   - \(a = x \Leftrightarrow b = y\) (one-to-one condition)
   - \(a\) is to the left of \(x\) \(\Rightarrow\) \(b\) is to the left of \(y\) (order condition)
   - \(a\) is an ancestor of \(x\) \(\Rightarrow\) \(b\) is an ancestor of \(y\) (ancestor condition)
3. Optional: \(a = \text{root}(T_1)\) and \(b = \text{root}(T_2)\) \(\Rightarrow\) \((a, b) \in M\) (forbid deleting the root node)

\[^1\text{I.e., a precedes x in both preorder and postorder}\]

Constrained Edit Distance

- We compute a special case of the edit distance to get a faster algorithm.
- \(\text{lca}(a, b)\) is the lowest common ancestor of \(a\) and \(b\).
- Additional requirement on the mapping \( M \):
  - \(\text{lca}(a_1, a_2)\) is a proper ancestor of \(x\) \(\Leftrightarrow\) \(\text{lca}(b_1, b_2)\) is a proper ancestor of \(y\).
- Intuition: Distinct subtrees of \(T_1\) are mapped to distinct subtrees of \(T_2\).
Example: Constrained Edit Distance

\[ T_1 \]
\[ a \]
\[ b \quad c \quad d \]
\[ e \quad f \quad g \]
\[ T_2 \]
\[ a \]
\[ b \quad c \quad d \]
\[ e \quad f \quad g \]

- **Constrained** edit distance (dashed lines): \( \delta_c(T_1, T_2) = 5 \)
  - constrained mapping \( M_c = \{(a,a), (d,d), (c,i), (f,f)(g,g)\} \)
  - edit sequence: \( \text{ren}(c,i), \text{del}(b), \text{del}(e), \text{ins}(h), \text{ins}(e) \)
- **Unconstrained** edit distance (dotted lines): \( \delta_t(T_1, T_2) = 3 \)
  - mapping \( M_t = \{(a,a), (d,d), (e,e), (c,i), (f,f)(g,g)\} \)
  - edit sequence: \( \text{ren}(c,i), \text{del}(b), \text{ins}(h) \)

\( (e,e) \) violates the 4th condition of the constrained mapping:
- \( \text{lca}(e,f) \) in \( T_1 \) is a
- \( a \) is a proper ancestor of \( d \) in \( T_1 \)
- assume \( (e,e), (f,f), (d,d) \in M_c \)
- \( \text{lca}(e,f) \) in \( T_2 \) is \( h \)
- \( h \) is not a proper ancestor of \( d \) in \( T_2 \)

**Theorem (Complexity of the Constrained Edit Distance)**

Let \( T_1 \) and \( T_2 \) be two trees with \( |T_1| \) and \( |T_2| \) nodes, respectively. There is an algorithm that computes the constrained edit distance between \( T_1 \) and \( T_2 \) with runtime \( O(|T_1||T_2|) \).

**Proof.**
See [Zha95, GJK+02].

**Theorem (Upper Bound)**

Let \( T_1 \) and \( T_2 \) be two trees, let \( \delta_t(T_1, T_2) \) be the unconstrained and \( \delta_c(T_1, T_2) \) be the constrained tree edit distance, respectively. Then

\[ \delta_t(T_1, T_2) \leq \delta_c(T_1, T_2) \]

**Proof.**
See [GJK+02].
The constrained edit distance can be computed faster:
- constrained edit distance runtime: $O(n^2)$
- unconstrained edit distance runtime: $O(n^3)$

Similarity join: match all trees with $\delta_T(T_1, T_2) \leq \tau$
- if $\delta_c(T_1, T_2) \leq \tau$ then also $\delta_T(T_1, T_2) \leq \tau$.
- thus we do not have to compute the expensive tree edit distance

Sudipto Guha, H. V. Jagadish, Nick Koudas, Divesh Srivastava, and Ting Yu.
Approximate XML joins.

Kaizhong Zhang.
Algorithms for the constrained editing distance between ordered labeled trees and related problems.