Similarity search on tree structured data

Ziying Pan (Cedar)

University of Illinois at Urbana–Champaign

March 15, 2011
Efficient Similarity Search for Hierarchical Data in Large Databases (EBDT04’)

Karin Kailing, Hans-Peter Kriegel, Stefan Schönonauer, and Thomas Seidl

University of Munich, Institute for Computer Science.
Introduction to tree structured data

- Complex objects often carry some kind of hierarchically internal structure.
- Tree is a data structure representing hierarchical nature of information about an object.
- Tree structure is the most simple and intuitive way to visualize the information of a complex object.
Examples of tree structured data

Examples: documents, images, chemical compounds, XML data, etc.
Question
How can we classify or cluster among tree structured data?

A step further...

Question
How can we measure similarity among tree structured data?
**General philosophy**: measuring similarity of trees based on the cost of tree transformation, i.e., the minimal number of edit operations necessary to transform one tree to into the other.

**Effectiveness**: taking into account both structural and content-based information of the trees.
ED—types of basic edit operations in tree transformation

- **Insertion**: Inserting a node $n$ in a tree below a node $p$ means that $p$ becomes the parent of $n$ and a subset of $p$’s children becomes the $n$’s children.

- **Deletion**: the inverse operation of insertion.

- **Relabeling**: change the label of a node.
The edit distance between two trees $t_1$ and $t_2$, $ED(t_1, t_2)$, is the minimum cost of all edit sequences that transform $t_1$ to $t_2$:

$$ED(t_1, t_2) = \min\{c(S) | S \text{ a seq of edit operations transforming } t_1 \text{ into } t_2\}$$

**Remark:** Such a complex measure is not suitable for large data sets. Hence a constrained edit distance measure is proposed by Zhang et al.

The degree 2 edit distance is the same as edit distance, except that only insertions or deletions of nodes with maximum number of two edges are allowed.

**Remark:** While degree 2 ED has a polynomial time complexity, it is still too complicated for the use in large databases.
Main result: Filter Refinement Architecture (FRA)

- **Core idea**: Obtain a small set of candidates to a query by applying a filter criterion to the database objects.

- **Effect**: To reduce the number of expensive similarity distance calculations and speed up the search process.

![Diagram](image)

**Fig. 1.** The filter-refinement architecture.
More on FRA

**Approach:** Approximate the edit distance using using a lower bounding function.

**Question**

How can we find such lower bounding functions that can become good filters?

We can look for them by examining the **features** of trees:

- Depth of the tree -> Height of the nodes in the tree
- Width of the tree -> Degrees of nodes in the tree.
- Content of the tree -> Labels of the nodes.
Idea: Project the data trees into the height feature subspace.

Definition

Leaf distance of a node \( n, d_l(n) \), is the maximal length of a path from \( n \) to any leaf node the subtree rooted at \( n \).

Definition

Leaf distance histogram: a vector where each coordinate (bin) is the number of nodes that share the same leaf distance.

\[
    h_l(t)[i] = |n \in t, d_l(n) = i|, \text{where } 0 \leq i \leq \text{height}(t)
\]
For any two trees $t_1$ and $t_2$, the $L_1$-distance of the leaf distance histograms is a lower bound of the degree-2 edit distance of $t_1$ and $t_2$:

$$L_1(h_l(t_1), h_l(t_2)) \leq ED_2(t_1, t_2).$$

Hence, $L_1(h_l(t_1), h_l(t_2))$ is a desired lower bound function of ED.
FRA—Filtering based on degrees of nodes

**Idea:** Project the data trees into node-degree feature subspace to seek a lower bound of ED.

**Approach:** Construct the degree histogram for the nodes and compare.

**Definition**

Degree histogram $h_d(t)$: a vector where each coordinate (bin) is the number of nodes that share the same degree.

i.e, $h_d(t)[i] = |n \in t, degree(n) = i|$, where $0 \leq i \leq degree_{max}(t)$.
Theorem

For any two trees $t_1$ and $t_2$, the $L_1$-distance of the degree histograms divided by three is a lower bound of the edit distance of $t_1$ and $t_2$:

$$\frac{L_1(h_d(t_1), h_d(t_2))}{3} \leq ED(t_1, t_2).$$

Hence, $L_1(h_d(t_1), h_d(t_2))$ is a desired lower bound function of ED.
Filtering based on labels of nodes

• Node labels can be used to design filters since content features are expressed in them.

• **Idea:** Use the difference of distribution values of labels to obtain a lower bound of editing distance.

• **Approach:** Depending on label distributions
  - **Discrete:** Use histogram to approximate distribution values.
  - **Continuous:** Use a continuous weight function to approximate distribution values.
Filtering based on labels of nodes: discrete case

**Label histogram:** divide node label into several bins and each bin is assigned the number of nodes that are in the range of the bin.

**Approach:** Use half of the $L_1$-distance of label histogram to approximate ED.

**Reason for half:** A single relabeling operation can influence at most two bins.
Filtering based on labels of nodes: continuous case

- **Feature value function:**

  \[
  f(t) = \sum_{i=1}^{\mid t \mid} \mid x_i \mid
  \]

  where \( x_i \) is the \( i \)th node label value of the tree \( t \), and \( \mid t \mid \) is the number of the nodes in the tree \( t \).

- **Filter function:**

  \[
  d_{\text{filter}}(t_1, t_2) = \frac{\mid f(t_1) - f(t_2) \mid}{\text{max}_{\text{diff}}}
  \]

  where \( \text{max}_{\text{diff}} \) is the maximal possible difference between two attribute values, i.e., \( \text{max}_{i,j} \mid x_i - x_j \mid \).
An example for continuous case

Fig  Filtering for continuous weight functions.
Table 1. Statistics of the data set.

<table>
<thead>
<tr>
<th></th>
<th>number of images</th>
<th>number of nodes</th>
<th>height</th>
<th>maximal degree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>max</td>
<td>min</td>
<td>Ø</td>
<td>max</td>
</tr>
<tr>
<td>commercial color images</td>
<td>8,536</td>
<td>331</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>color TV-images</td>
<td>43,000</td>
<td>109</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>black and white pictographs</td>
<td>705</td>
<td>113</td>
<td>3</td>
<td>13</td>
</tr>
</tbody>
</table>

Fig. 9. Runtime and number of candidates for k-nn-queries on 10,000 color TV-images.
Thank you!

Questions?