Similarity Search
The String Edit Distance

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

Unit 2 – March 8, 2012

Outline
String Edit Distance
  • Motivation and Definition
  • Brute Force Algorithm
  • Dynamic Programming Algorithm
  • Edit Distance Variants

Motivation
• How different are
  • hello and hello?
  • hello and hallo?
  • hello and hell?
  • hello and shell?
What is a String Distance Function?

Definition (String Distance Function)

Given a finite alphabet \( \Sigma \), a **string distance function**, \( \delta \), maps each pair of strings \((x, y) \in \Sigma^* \times \Sigma^*\) to a positive real number (including zero).

\[
\delta : \Sigma^* \times \Sigma^* \rightarrow \mathbb{R}^+_0
\]

- \( \Sigma^* \) is the set of all strings over \( \Sigma \), including the **empty string** \( \varepsilon \).

The String Edit Distance

Definition (String Edit Distance)

The **string edit distance** between two strings, \( \text{ed}(x, y) \), is the minimum number of character insertions, deletions and replacements that transforms \( x \) to \( y \).

- Example:
  - \( \text{hello} \rightarrow \text{hallo} \): replace \( e \) by \( a \)
  - \( \text{hello} \rightarrow \text{hell} \): delete \( o \)
  - \( \text{hello} \rightarrow \text{shell} \): delete \( o \), insert \( s \)

Also called **Levenshtein distance**.\(^1\)

---

\(^1\)Levenshtein introduced this distance for signal processing in 1965 [Lev65].

Gap Representation

- **Gap representation** of the string transformation \( x \rightarrow y \):
  - Place string \( x \) above string \( y \)
  - with a gap in \( x \) for every insertion,
  - with a gap in \( y \) for every deletion,
  - with different characters in \( x \) and \( y \) for every replacement.

Any sequence of edit operations can be represented with gaps.

- **Example**:
  - \( \text{hallo} \)
  - \( \text{shell} \)
  - insert \( s \)
  - replace \( a \) by \( e \)
  - delete \( o \)
Brute Force Algorithm

Example:

```
hallo
shell
```

Given: Gap representation, \(gap(x, y)\), of the shortest edit distance between two strings \(x\) and \(y\), such that \(gap(x, y) = ed(x, y)\).

Claim:
- If we remove the last column, then the remaining columns represent the shortest edit distance, \(gap(x', y') = ed(x', y')\), between the remaining substrings, \(x'\) and \(y'\).

Proof (by contradiction):
- Last column contributes with \(c = 0\) or \(c = 1\) to \(gap(x, y)\), thus \(gap(x, y) = gap(x', y') + c\).
- If we assume \(ed(x', y') < gap(x', y')\), then we could find a new gap representation \(gap^*(x', y') = ed(x', y') < gap(x', y')\) such that \(gap^*(x, y) = gap^*(x', y') + c < gap(x', y') + c = ed(x, y)\).

\[\square\]

**Brute Force Algorithm**

\[ed-bf(x, y)\]

\[m = |x|, \ n = |y|\]

\[\text{if} \ m = 0 \ \text{then return} \ n\]

\[\text{if} \ n = 0 \ \text{then return} \ m\]

\[\text{if} \ x[m] = y[n] \ \text{then} \ c = 0 \ \text{else} \ c = 1\]

\[\text{return} \ \min(ed-bf(x, y[1 \ldots n - 1]) + 1, \ ed-bf(x[1 \ldots m - 1], y) + 1, \ ed-bf(x[1 \ldots m - 1], y[1 \ldots n - 1]) + c)\]

\[\text{Exponential runtime in string length} \ :-(\]

**Observation**: Subproblems are computed repeatedly (e.g. \(ed-bf(a, x)\) is computed 3 times)

**Approach**: Reuse previously computed results!
Dynamic Programming Algorithm

- **Store distances** between all prefixes of $x$ and $y$
- Use matrix $C_{0..m,0..n}$ with
  \[
  C_{i,j} = ed(x[1..i], y[1..j])
  \]
  where $x[1..0] = y[1..0] = \varepsilon$.

- **Example**:

  \[
  \begin{array}{c|cccc}
  & \varepsilon & x & a & b \\
  \hline
  \varepsilon & 0 & 1 & 2 & \vdots \\
  a & 1 & 1 & 2 & \vdots \\
  b & 2 & 2 & 1 & 1 \\
  \end{array}
  \]

Understanding the Solution

- **Example**:

  \[
  \begin{array}{c|cccc}
  & \varepsilon & m & o & n & d \\
  \hline
  \varepsilon & 0 & 1 & 2 & 3 & 4 \\
  m & 1 & 0 & 1 & 2 & 3 \\
  o & 2 & 1 & 0 & 1 & 2 \\
  n & 3 & 2 & 1 & 1 & 2 \\
  \end{array}
  \]

  \[
  \begin{array}{c|cccc}
  & \varepsilon & m & o & n & d \\
  \hline
  \varepsilon & 0 & 1 & 2 & 3 & 4 \\
  m & 1 & 0 & 1 & 2 & 3 \\
  o & 2 & 1 & 0 & 1 & 2 \\
  n & 3 & 2 & 1 & 1 & 2 \\
  \end{array}
  \]

  - **Solution 1**: replace $n$ by $d$ and (second) $o$ by $n$ in $x$
  - **Solution 2**: insert $d$ after $n$ and delete (first) $o$ in $x$
  - **Solution 3**: insert $d$ after $n$ and delete (second) $o$ in $x$
Dynamic Programming Algorithm – Properties

- **Complexity:**
  - $O(mn)$ time (nested for-loop)
  - $O(mn)$ space (the $(m+1) \times (n+1)$-matrix $C$)

- **Improving space complexity** (assume $m < n$):
  - we need only the previous column to compute the next column
  - we can forget all other columns
  - $\Rightarrow O(m)$ space complexity

### String Edit Distance Dynamic Programming Algorithm

```
ed-dyn+(x, y)
col0 : array[0..|x|]
col1 : array[0..|x|]
for i = 0 to |x| do col0[i] = i
for j = 1 to |y| do
  col1[0] = j
  for i = 1 to |x| do
    if x[i] = y[j] then c = 0 else c = 1
    col1[i] = min(col0[i - 1] + c,
                   col1[i - 1] + 1,
                   col0[i] + 1)
  col0 = col1
```

### Distance Metric

**Definition (Distance Metric)**

A distance function $\delta$ is a **distance metric** if and only if for any $x, y, z$ the following hold:

- $\delta(x, y) = 0 \iff x = y$ (identity)
- $\delta(x, y) = \delta(y, x)$ (symmetric)
- $\delta(x, y) + \delta(y, z) \geq \delta(x, z)$ (triangle inequality)

**Examples:**

- the Euclidean distance is a metric
- $d(a, b) = a - b$ is not a metric (not symmetric)
Introducing Weights

- Look at the edit operations as a set of rules with a cost:
  \[
  \begin{align*}
  \alpha(\varepsilon, b) &= \omega_{\text{ins}} \\
  \alpha(a, \varepsilon) &= \omega_{\text{del}} \\
  \alpha(a, b) &= \begin{cases}
  \omega_{\text{rep}} & \text{if } a \neq b \\
  0 & \text{if } a = b
  \end{cases}
  \end{align*}
  \]
  where \(a, b \in \Sigma\), and \(\omega_{\text{ins}}, \omega_{\text{del}}, \omega_{\text{rep}} \in \mathbb{R}_0^+\).

  - Edit script: sequence of rules that transform \(x\) to \(y\)
  - Edit distance: sequence of rules with minimum cost (adding up costs of single rules)
  - Example: so far we assumed \(\omega_{\text{ins}} = \omega_{\text{del}} = \omega_{\text{rep}} = 1\).

Variants of the Edit Distance

- Unit cost edit distance (what we did so far):
  \(\omega_{\text{ins}} = \omega_{\text{del}} = \omega_{\text{rep}} = 1\)
  \(0 \leq ed(x, y) \leq \max(|x|, |y|)\)
  distance metric

  - Hamming distance [Ham50, SK83]:
    - called also “string matching with \(k\) mismatches”
    - allows only replacements
    - \(\omega_{\text{rep}} = 1, \omega_{\text{ins}} = \omega_{\text{del}} = \infty\)
    \(0 \leq d(x, y) \leq |x|\) if \(|x| = |y|\), otherwise \(d(x, y) = \infty\)
  distance metric

  - Longest Common Subsequence (LCS) distance [NW70, AG87]:
    - allows only insertions and deletions
    - \(\omega_{\text{ins}} = \omega_{\text{del}} = 1, \omega_{\text{rep}} = \infty\)
    \(0 \leq d(x, y) \leq |x| + |y|\)
  distance metric

  \(LCS(x, y) = (|x| + |y| - d(x, y))/2\)

Weighted Edit Distance

- Recursive formula with weights:
  \[
  \begin{align*}
  C_{0,0} &= 0 \\
  C_{i,j} &= \min(C_{i-1,j-1} + \alpha(x[i], y[j]), \\
  &\quad C_{i-1,j} + \alpha(x[i], \varepsilon), \\
  &\quad C_{i,j-1} + \alpha(\varepsilon, y[j]))
  \end{align*}
  \]
  where \(\alpha(a, a) = 0\) for all \(a \in \Sigma\), and \(C_{-1,j} = C_{i,-1} = \infty\).

- We can easily adapt the dynamic programming algorithm.

Allowing Transposition

- Transpositions
  - switch two adjacent characters
  - can be simulated by delete and insert
  - typos are often transpositions

  - New rule for transposition
    \[
    \alpha(ab, ba) = \omega_{\text{trans}}
    \]
    allows us to assign a weight different from \(\omega_{\text{ins}} + \omega_{\text{del}}\)

  - Recursive formula that includes transposition:
    \[
    \begin{align*}
    C_{0,0} &= 0 \\
    C_{i,j} &= \min(C_{i-1,j-1} + \alpha(x[i], y[j]), \\
    &\quad C_{i-1,j} + \alpha(x[i], \varepsilon), \\
    &\quad C_{i,j-1} + \alpha(\varepsilon, y[j]), \\
    &\quad C_{i-2,j-2} + \alpha(x[i-1]x[i], y[j-1]y[j]))
    \end{align*}
    \]
    where \(\alpha(ab, cd) = \infty\) if \(a \neq d\) or \(b \neq c\), \(\alpha(a, a) = 0\) for all \(a \in \Sigma\),
    and \(C_{-1,j} = C_{i,-1} = C_{-2,j} = C_{j,-2} = \infty\).
Example: Edit Distance with Transposition

**Example:** Compute distance between $x = \text{meal}$ and $y = \text{mael}$ using the edit distance with transposition ($\omega_{\text{ins}} = \omega_{\text{del}} = \omega_{\text{rep}} = \omega_{\text{trans}} = 1$)

<table>
<thead>
<tr>
<th></th>
<th>m</th>
<th>a</th>
<th>e</th>
<th>l</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>m</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>l</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

The value in red results from the transposition of $ea$ to $ae$.

Example: Text Searching

**Example:**

- $p = \text{survey}$
- $t = \text{surgery}$
- $k = 2$

<table>
<thead>
<tr>
<th></th>
<th>s</th>
<th>u</th>
<th>r</th>
<th>g</th>
<th>e</th>
<th>r</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$p$ = survey</td>
<td>s</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t$ = surgery</td>
<td>u</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$k$ = 2</td>
<td>r</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>v</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

**Solutions:** 3 matching positions with $k \leq 2$ found.

- survey
- surge
- survey
- surger
- survey
- surgery


David Sankoff and Josef B. Kruskal, editors.