Creative Computing II

Christophe Rhodes
c.rhodes@gold.ac.uk

Autumn 2010, Wednesdays:
10:00–12:00: RHB307 & 14:00–16:00: WB316
Winter 2011, Wednesdays:
10:00–12:00: RHB307 & 14:00–16:00: WB316

Levenshtein distance:

- define a set of permitted operations and associated costs:
  - insert (cost *ins*);
  - delete (cost *del*);
  - substitute (cost *sub*);
- Levenshtein distance between two words is the minimum cost to transform one word into another.

$d \leftarrow d_{Levenshtein}(x, y)$

  $l_x \leftarrow \text{length}(x)$; $l_y \leftarrow \text{length}(y)$
  if $l_x = 0$ **then**
    $d \leftarrow l_y$
  else if $l_y = 0$ **then**
    $d \leftarrow l_x$
  else
    $d_{del} \leftarrow del + d_{Levenshtein}(x_{2:l_x}, y)$
    $d_{ins} \leftarrow ins + d_{Levenshtein}(x, y_{2:l_y})$
    if $x_1 = y_1$ **then**
      $d_{sub} \leftarrow d_{Levenshtein}(x_{2:l_x}, y_{2:l_y})$
    else
      $d_{sub} \leftarrow sub + d_{Levenshtein}(x_{2:l_x}, y_{2:l_y})$
    end if
    $d \leftarrow min(d_{del}, d_{ins}, d_{sub})$
  end if

This computation is $O(L^L)$ for strings of length L

$d \leftarrow d_{Levenshtein}(x, y)$
    $l_x \leftarrow \text{length}(x)$; $l_y \leftarrow \text{length}(y)$
    if $l_x = 0$ **then**
        $d \leftarrow l_y$
    else if $l_y = 0$ **then**
        $d \leftarrow l_x$
    else
        $d_{del} \leftarrow del + d_{Levenshtein}(x_{2:l_x}, y)$
        $d_{ins} \leftarrow ins + d_{Levenshtein}(x, y_{2:l_y})$
        if $x_1 = y_1$ **then**
            $d_{sub} \leftarrow d_{Levenshtein}(x_{2:l_x}, y_{2:l_y})$
        else
            $d_{sub} \leftarrow sub + d_{Levenshtein}(x_{2:l_x}, y_{2:l_y})$
        end if
        $d \leftarrow min(d_{del}, d_{ins}, d_{sub})$
    end if

This computation is $O(L^L)$ for strings of length L … but we can do better: "Dynamic Programming".

Multimedia Information Retrieval

Textual Distance Measures

$d \leftarrow d_{Levenshtein}(x, y)$

```
for i from 0 to l_x do
    d_{i,0} ← i × del
end for
for j from 0 to l_y do
    d_{0,j} ← j × ins
end for
for i from 1 to l_x do
    for j from 1 to l_y do
        if x_i = y_j then
            d_{i,j} = d_{i-1,j-1}
        else
            d_{i,j} = min(d_{i-1,j} + del, d_{i,j-1} + ins, d_{i-1,j-1} + sub)
        end if
    end for
end for
d ← d_{l_x,l_y}
```

This computation is $O(L^2)$ for strings of length L.

$$d(\texttt{choose}, \texttt{choose}) = 0$$

$$d(\texttt{choose}, \texttt{chose}) = \mathit{del}$$

$$d(\texttt{choose}, \texttt{chives}) = 2 \times \mathit{sub} + \mathit{del} + \mathit{ins}$$

$$d(\texttt{professor}, \texttt{proffessor}) = \mathit{ins}$$

$$d(\texttt{professors}, \texttt{proffessor}) = \mathit{ins} + \mathit{del}$$

- Often an appropriate measure to use for comparing words;
- Models ways of making mistakes;
- $O(L^2)$ time is practical for distances between words (but not between whole documents).

Term-Frequency–Inverse-Document-Frequency (**tf-idf**):

- intuition:
  - term frequency: the more often a term is in a document, the more relevant it is;
  - inverse document frequency: the more documents a term is in, the less discriminating it is;
- Therefore, maximize a measure combining the term frequency and the inverse document frequency.

Term-Frequency–Inverse-Document-Frequency (**tf-idf**):

- intuition:
  - term frequency: the more often a term is in a document, the more relevant it is;
  - inverse document frequency: the more documents a term is in, the less discriminating it is;
- Therefore, maximize a measure combining the term frequency and the inverse document frequency.
- $\mathrm{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}$

Term-Frequency–Inverse-Document-Frequency (**tf-idf**):

- intuition:
  - term frequency: the more often a term is in a document, the more relevant it is;
  - inverse document frequency: the more documents a term is in, the less discriminating it is;
- Therefore, maximize a measure combining the term frequency and the inverse document frequency.
- $\mathrm{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}$
- $\mathrm{idf}_i = \log \frac{|D|}{|d_j : n_{ij} > 0|}$