String-searching Algorithms

Levenshtein distance

Sergio Peignier sergio.peignier@insa-lyon.fr

Associate Professor
INSA Lyon
Biosciences department

Hamming distance | summary

Let $\{S, T\} \in \mathcal{S}^2$ s.t. $|S| = |T|$

Hamming distance:

$d(T, S) = |\{i \quad \forall i \in \{1, ..., |S|\} \mid T[i] \neq S[i]\}.$

Number of positions where both strings differ

Pros:

- Simple distance
- Easy and fast to compute

Cons:

- Only works for $|S| = |T|$
- Does not accurately reals reflects strings similarity.

Hamming distance | Example

$S =$ ”dans l’herbe noire les kobolds vont”

$T =$ ”dans l’herbr noird lzs kobolds vont”

$$d(S, T) = 3$$

Hamming distance | Example

$S =$ ”dans l’herbe noire les kobolds vont ”

$T =$ ” dans l’herbe noire les kobolds vont”

$$d(S, T) = 36$$

Levenshtein distance

Goal:

- Compare strings with different lengths.
- Works for strings that underwent common modification (e.g., insertions, deletions)

Levenshtein distance

- Let $\{S, T\} \in \mathcal{S}^2$ possibly $|S| \neq |T|$
- Let $\mathbb{1}_{S[i] \neq T[j]}$ be the indicator function
  $\mathbb{1}_{S[i] \neq T[j]} = 1$ if $S[i] \neq T[j]$ and 0 otherwise.
- $\mathcal{L}_{S,T}(i, j)$: Levenshtein dist. between $S[1, i]$ and $T[1, j]$

Levenshtein distance definition

if $min(i, j) = 0$: $\quad \mathcal{L}_{S,T}(i, j) = max(i, j)$

Otherwise:

$$\mathcal{L}_{S,T}(i, j) = min \begin{cases} \mathcal{L}_{S,T}(i-1, j) + 1 \\ \mathcal{L}_{S,T}(i, j-1) + 1 \\ \mathcal{L}_{S,T}(i-1, j-1) + \mathbb{1}_{S[i] \neq T[j]} \end{cases}$$

Wagner–Fischer algorithm

- The **Wagner–Fischer** algorithm allows to compute the Levenshtein distance
- **Dynamic programming** algorithm
- Compute a matrix with the Levenshtein distances between all prefixes of both strings
- Value at row $|S|$ and column $|T|$ stores the Levenshtein distance

Example