Comparaison de mots : distance d'édition

Manon Ruffini

Soient x et y deux mots sur un alphabet Σ. On voudrait définir une distance entre ces mots.

Définition 1

Les opérations d'éditions sont les opérations suivantes :

- *La substitution d'une lettre a de x à une position donnée par une lettre b de y. On note :* $\binom{a}{b}$
- *La suppression d'une lettre a de x à une position donnée. On note* $\binom{a}{-}$
- *L'insertion d'une lettre de y dans x à une position donnée. On note* $\binom{-}{a}$

On associe un coût à chacune de ces opérations : $\forall a, b \in \Sigma$, on note :

- $\mathrm{Sub}(a, b)$ le coût de la substitution de a par b
- $\mathrm{Supp}(a)$ le coût de la suppression de a
- $\mathrm{Ins}(a)$ le coût de l'insertion de a

Définition 2

On définit la distance d'édition entre x et y comme le coût d'une suite d'opérations d'éditions de coût minimal, qui transforme x en y :

$$Lev(x, y) = \min\{coût\ de\ \sigma | \sigma \in \Sigma_{x,y}\}$$

où $\Sigma_{x,y}$ est l'ensemble des suites d'opérations d'éditions qui transforment x en y

[1]

But : Calculer la distance d'édition.

Idée : utiliser un graphe d'édition : Les sommets sont les couples (lettre de $x\times$ lettre de y). Flèches : $\rightarrow$ = suppression, $\downarrow$ = insertion, $\searrow$ = substitution. Les flèches sont étiquetées par le coût des opérations auxquelles elles correspondent.

Exemple 1

Faire un exemple (cf Crochemore)

On veut résoudre le problème suivant :

Entrée : Un mot x de taille m ; un mot y de taille n.

Sortie : La distance d'édition entre x et y

Et pour ça, on va utiliser la programmation dynamique.

- Sous problèmes : $\forall 0 \leqslant i \leqslant m$ et $\forall 0 \leqslant j \leqslant n$, $T[i, j] = \mathrm{Lev}(x[1, i], y[1, j])$
- Relations de récurrence : pour $1 \leqslant i \leqslant m$ et $1 \leqslant j \leqslant j$

$$T[0, 0] = 0$$
$$T[i, 0] = T[i-1, 0] + \mathrm{Supp}(x[i])$$
$$T[0, j] = T[j-1, 0] + \mathrm{Ins}(y[j])$$

1. Lev est une distance sur Σ^* ssi Sub est une distance sur Σ^2 et $\forall a \in \Sigma, \mathrm{Supp}(a) = \mathrm{Ins}(a) > 0$

$$T[i,j] = \min \begin{cases} T[i-1,j-1] + \mathrm{Sub}(x[i], y[j]) \\ T[i-1,j] + Supp(x[i]) \\ T[i,j-1] + Ins(y[j]) \end{cases}$$

Lemme 1

Pour tous $a, b \in A, u, v \in A^*$, on :

- $Lev(ua, \varepsilon) = Lev(u, \varepsilon) + Supp(a)$
- $Lev(\varepsilon, vb) = Lev(\varepsilon, v) + Ins(b)$
- $Lev(ua, vb) = \min \begin{cases} Lev(u, v) + Sub(a, b) \\ Lev(u, vb) + Supp(a) \\ Lev(ua, v) + Ins(b) \end{cases}$

Démonstration : Les suites des opérations qui transforment ua en ε se terminent nécessairement par la suppression de la lettre a. Le reste de la suite transforme u en le mot vide. D'où :

$$\begin{aligned} \mathrm{Lev}(ua, \varepsilon) &= \min\{\text{coût de } \sigma, \sigma \in \Sigma_{ua,\varepsilon}\} \\ &= \min\{\text{coût de } \sigma' \cdot \binom{a}{\varepsilon}, \sigma' \in \Sigma_{u,\varepsilon}\} \\ &= \min\{\text{coût de } \sigma', \sigma \in \Sigma_{u,\varepsilon}\} + \mathrm{Supp}(a) \\ &= \mathrm{Lev}(u, \varepsilon) + \mathrm{Supp}(a) \end{aligned}$$

La deuxième identité se montre de la même façon.
Pour la troisième identité, il faut distinguer les cas où la dernière opération est une substitution, une suppression ou une insertion. ■

Ainsi, la relation de récurrence précédente vient du fait que $\mathrm{Lev}(\varepsilon, \varepsilon) = 0$ et du lemme avec $a = x[i], b = y[j], u = x[1, i-1]$ et $v = y[1, j-1]$.

On obtient donc l'algorithme suivant :

Algorithme 1 : Calcul-Lev(x, y)

```
T[0..m, 0..n] ← tableau;
T[0,0] ← 0;
pour 1 ⩽ i ⩽ m faire
    T[i,0] ← T[i-1,0] + Supp(x[i])
fin
pour 1 ⩽ j ⩽ n faire
    T[0,j] = T[j-1,0] + Ins(y[j]);
    pour 1 ⩽ i ⩽ m faire
        T[i,j] = min { T[i-1,j-1] + Sub(x[i], y[j])
                     { T[i-1,j] + Supp(x[i])
                     { T[i,j-1] + Ins(y[j])
    fin
fin
retourner T[m,n]
```

Cet algorithme s'exécute en temps $O(m \times n)$ dans un espace $O(\min(m, n))$. En effet, le calcul de chaque position de T s'effectue en temps constant : on initialise en $O(m+n)$ puis on calcule les $m \times n$ autres valeurs de T. De plus, on remarque que seules deux lignes ou deux colonnes de T suffisent pour réaliser le calcul.

Calcul d'un alignement optimal

L'algorithme précédent calcule la distance d'édition, mais ne renvoie pas de suite d'opération optimale. Pour en trouver une, on a besoin de garder en mémoire toute la table T. Ensuite, on remonte dans la table à partir de $T[m, n]$ de la façon suivante :
A partir d'une position $[i, j]$, on visite, parmi les trois positions voisines $[i-1, j-1], [i, j-1], [i-1, j]$, l'une de celle dont la valeur produit celle de $T[i, j]$.

Références

[1] M. Crochemore, C. Hancart, T. Lecroq, *Algorithmique du texte*. Vuibert, 2001.