Méthodes et outils pour la biologie des systèmes
Projet Algorithmique et Structures de Données Avancées

Alain Griffault

Lundi 10 mars 2014

La distance d'édition

Définition et idées

La distance d'édition est une distance entre deux mots qui reflète le coût minimal du remplacement du premier mot par le second dans un texte.

En imaginant pour chacun des deux mots, un curseur avançant lettre à lettre ; ce coût s'exprime en fonction de quelques opérations élémentaires sur un mot.

- si les deux curseurs pointent sur deux lettres identiques, avancer les deux curseurs correspond à **copier dans le second mot une lettre du premier mot.**
- si les deux curseurs pointent sur deux lettres différentes, avancer les deux curseurs correspond à **remplacer une lettre du second mot par une lettre du premier mot**.
- avancer seulement le curseur du premier mot correspond à **supprimer une lettre du premier mot**.
- avancer seulement le curseur du second mot correspond à **insérer une lettre du second mot**.

Notons :

- `x` l'opération de **copier dans le second mot une lettre du premier mot**, et c_c le coût associé.
- `(x|y)` l'opération de **remplacer une lettre du second mot par une lettre du premier mot**, et c_r le coût associé.
- `(-x)` l'opération de **supprimer une lettre du premier mot**, et c_s le coût associé.
- `(+x)` l'opération de **insérer une lettre du second mot**, et c_i le coût associé.

Ainsi les séquences d'opérations :

- `tr1(totobbbi,atiti) = (+a)(+t)(+i)(+t)+(i)(-t)(-o)(-t)(-o)(-b)(-b)(-b)(-i)` transforme le mot `totobbbi` en `atiti` pour un coût de $5 \times c_i + 8 \times c_s$
- `tr2(totobbbi,atiti) = (+a)t(o|i)t(-o)(-b)(-b)(-b)i` transforme le mot `totobbbi` en `atiti` pour un coût de $3 \times c_c + 1 \times c_r + 4 \times c_s + 1 \times c_i$

En général, l'opération de copier est gratuite, les opérations d'insertion et de suppression ont des coûts similaires, et l'opération de remplacement est plus onéreuse, mais moins que la somme d'une suppression suivi d'une insertion. Des valeurs typiques sont $c_c = 0, c_s = 2, c_i = 2, c_r = 3$. Avec ces valeurs, on obtient $cout(tr1(a,b)) = 26$ et $cout(tr2(a,b)) = 13$.

Définition : Soient a et b deux mots, la distance d'édition de a à b notée $de(a,b)$ est définie par

$$de(a,b) = minimum\{cout(tr(a,b))\}$$

Propriétés de la distance d'édition

Notation : Soit a un mot. On note a_i le mot composé des i premières lettres du mot a, et par convention a_0 désigne le mot vide.

Propriété : Soient a et b deux mots, la distance d'édition de a_i à b_j peut être calculée récursivement par :

$$\begin{cases} de(a_i,b_j) = j \times c_i & \text{si } i = 0 \\ de(a_i,b_j) = i \times c_s & \text{si } j = 0 \\ de(a_i,b_j) = min \begin{cases} de(a_{i-1},b_{j-1}) + c_c & \text{si } a[i] = b[j] \\ de(a_{i-1},b_{j-1}) + c_r & \text{si } a[i] \neq b[j] \\ de(a_i,b_{j-1}) + c_i \\ de(a_{i-1},b_j) + c_s \end{cases} & \text{si } i > 0 \text{ et } j > 0 \end{cases}$$

Algorithme du calcul de la distance d'édition

Cette distance peut se calculer par un algorithme récursif basé sur la définition précédente, mais un tel algorithme est assez inefficace, car il effectue de nombreuses fois les mêmes calculs.

La plupart des implantations de cette distance utilise la programmation dite *dynamique*. Cela consiste à stocker dans des tableaux les résultats des calculs intermédiaires afin de les utiliser ultérieurement au lieu de les recalculer.

		0	1	2	3	4	5
		ϵ	a	t	i	t	i
0	ϵ	**0**	**2**	4	6	8	10
1	t	2	3	**2**	4	6	8
2	o	4	5	4	**5**	7	9
3	t	6	7	5	7	**5**	7
4	o	8	9	7	8	**7**	8
5	b	10	11	9	10	**9**	10
6	b	12	13	11	12	**11**	12
7	b	14	15	13	14	**13**	14
8	i	16	17	15	13	15	**13**

FIGURE 1 – Table des distances entre a_i et b_j

Algorithme du calcul d'une transformation optimale

Une fois la table des distances $de(a_i, b_j)$ connue, il est relativement facile de construire la suite des opérations transformant le mot a en b. Cela correspond à trouver le chemin rouge dans la table. Pour cela il suffit de :

1. Partir de $de(a, b)$ avec deux indices i et$j$ initialisés aux longueurs des mots.
2. Itérer, tant que ces indices sont strictement positifs, la recherche d'une des actions possibles.
   - supprimer : la case *au dessus* doit avoir la même valeur diminuée du coût c_s.
   - insérer : la case *à gauche* doit avoir la même valeur diminuée du coût c_i.
   - copier : les lettres sont identiques et la diagonale doit avoir la même valeur diminuée du coût c_c.
   - remplacer : les lettres sont différentes et la diagonale doit avoir la même valeur diminuée du coût c_r.

La distance entre totobbbi et atiti est de	13.
Une transformation possible est	(+a)t(o|i)t(-o)(-b)(-b)(-b)i

FIGURE 2 – Distance et une des transformations optimales entre a_i et b_j

Travail à réaliser

Vous devez implémenter en `python` :

1. Un algorithme `distanceEditionCodage(a,b)` qui retourne une table contenant les distances d'éditions entre tous les mots a_i et b_j.
2. Un algorithme `distanceEditionDecodage(a,b,code)` qui retourne une suite d'opérations élémentaires sur les mots, permettant de transformer le mot a en b.
3. Au choix, un algorithme parmi les suivants :
   - Un algorithme qui prends en entrée un mot et un ensemble de mots défini au préalable, et qui retourne le mot de l'ensemble le plus proche du mot, ainsi qu'une transformation optimale.
   - Un algorithme qui prends en entrée un mot, un entier positif et un ensemble de mots défini au préalable, et qui retourne tous les mots de l'ensemble dont la distance d'édition est inférieure ou égale à l'entier. Idéalement les mots retournés seront ordonnés en fonction de cette distance.

Modalités

- **Travail en binôme** (1 seul trinôme accepté)
- Le rapport comprends les sources `python` des algorithmes, et un document contenant d'une part une étude de la complexité des différents algorithmes ; d'autre part des résultats pertinents d'exécution de vos programmes. Il sera envoyé par mail à `Alain.Griffault@labri.fr`
- Date limite de remise du travail : **Lundi 31 mars 2014 à 24 heures**.