TP Partitionnement Hiérarchique Agglomératif (supplément)

M. Tami, T. Thonet,
E. Gaussier

L'objectif de ce TP est d'implémenter une version alternative de l'**algorithme de partitionnement hiérarchique agglomératif** étudié lors des deux séances précédentes. Contrairement à ce dernier, ici la version de l'algorithme est spécifiquement adaptée à la méthode d'agrégation du **lien unique**. En effet, l'algorithme des deux séances précédentes était plus générique.

Comme le précédent, cet algorithme opère en deux étapes :

1. Construction de la matrice de similarité S entre documents (et les structures de données utiles) ;
2. Construction du dendrogramme.

Et il retourne aussi la liste des fusions L qui ont été effectuées.

Pour cette séance, nous utiliserons à nouveau **la matrice de similarité fournie** :

```
import numpy as np
sim_mat = np.array([[10, 6, 0, 0, 0, 0, 0, 0, 0],
                    [6, 10, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 10, 5, 3, 3, 1, 1, 0],
                    [0, 0, 5, 10, 1, 2, 1, 1, 0],
                    [0, 0, 3, 1, 10, 4, 1, 2, 0],
                    [0, 0, 3, 2, 4, 10, 1, 4, 0],
                    [0, 0, 1, 1, 1, 1, 10, 1, 0],
                    [0, 0, 1, 1, 2, 4, 1, 10, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 10]])
```

Nous rappelons que cette matrice contient la similarité (basées sur des caractéristiques phonologiques) entre les 9 langues suivantes : arabe, hébreu, sanskrit, avestique, grec classique, latin, gotique, irlandais ancien et le turc – cet ordre correspond à celui adopté dans la matrice ci-dessus.

Dans l'algorithme défini ci-dessous, nous utilisons les notations suivantes :

- $P[i]$ correspond initialement au document le plus proche de d_i (et donc ultérieurement à la classe la plus proche de la classe i). Noter qu'à la différence de l'algorithme étudié précédemment, $P[i]$ n'est plus une liste de priorité mais correspond à un **unique élément** (document ou classe).
- $P[i]$.sim désigne la **similarité** entre la classe i et sa classe la plus similaire.

Entrée : $C = \{d_1, \ldots, d_N\}$ collection de documents
pour $i \in \{1, \ldots, N\}$ **faire**
  pour $j \in \{1, \ldots, N\}$ **faire**
    $S[i][j] = sim\,(d_i, d_j)$; # **Initialisation de la matrice de similarité**
    # $P[i]$ **contient ici seulement le doc. le plus proche de** i **et sa similarité**
    si $i \neq j$ *et* $P[i].sim < S[i][j]$ **alors**
      $P[i].\text{sim} = S[i][j]$;
      $P[i].\text{index} = j$;
    fin
  fin
  $I[i] = i$; # **Indicateur de classe active**
fin
$L \leftarrow \emptyset$;
Construction du dendrogramme
pour ***chaque itération*** $k \in \{1, \ldots, N-1\}$ **faire**
  $i_1 = \underset{i:I[i]=i}{\text{argmax}}\, P[i].\text{sim}$;
  $i_2 = I[P[i_1].\text{index}]$;
  $P[i_1].\text{sim} = -1$;
  L.append($[i_1, i_2]$) ; # **Ajout de** (i_1, i_2) **à l'ensemble des fusions**
  pour $i \in \{1, \ldots, N\}$ **faire**
    si $I[i] = i$ *et* $i \neq i_1$ *et* $i \neq i_2$ **alors**
      $S[i][i_1] \leftarrow max\{S[i][i_1], S[i][i_2]\}$; # **Méthode du lien unique**
      $S[i_1][i] \leftarrow max\{S[i][i_1], S[i][i_2]\}$;
    fin
    si $I[i] = i_2$ **alors**
      $I[i] = i_1$
    fin
    si $I[i] = i$ *et* $i \neq i_1$ *et* $S[i_1][i] > P[i_1].sim$ **alors**
      $P[i_1].\text{sim} = S[i_1][i]$;
      $P[i_1].\text{index} = i$;
    fin
  fin
fin
Sortie : la liste des fusions L

Algorithme 1 : Alternative d'algorithme de partitionnement hiérarchique agglomératif pour le lien unique.

- $P[i]$.index désigne l'indice de cette dernière.
- $I[i]$ correspond à l'**indice** de la classe active à laquelle le document d_i est affecté. Noter qu'à la différence de l'algorithme étudié précédemment, I ne correspond plus à des valeurs booléennes indiquant si la classe est active ou non.

Algorithme de partitionnement hiérarchique agglomératif par la méthode d'agrégation du lien unique. Implémentez en Python l'algorithme de partitionnement hiérarchique agglomératif décrit dans l'Algorithme 1 et testez-le sur la matrice de similarité donnée précédemment.

Tracez le dendrogramme associé au partitionnement hiérarchique obtenu. Vérifiez la cohérence des résultats.

Que pensez-vous de la complexité de l'Algorithme 1 par rapport à celui étudié lors des deux dernières séances ?