TP noté : Partitionnement hiérarchique agglomératif

M. Tami, T. Thonet,
E. Gaussier

L'objectif de ce TP est d'étudier un **algorithme de partitionnement hiérarchique agglomératif** qui est (tout comme l'algorithme des k-moyennes précédemment étudié) une approche standard de partitionnement des données (*clustering*). Ici, la méthode d'agrégation retenue est celle de la **méthode à lien unique**.

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.

Cet algorithme retourne la liste des fusions L qui ont été effectuées.

Pour cette séance, **la matrice de similarité vous est 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]])
```

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 :

- $P[i]$ désigne la file de priorité associée au document d_i de la collection (et donc associée à chaque classe i initiale). Les files de priorité sont mises à jour au fur et à mesure de la construction du dendrogramme.
- $P[i].MAX$.sim désigne la similarité maximum entre la classe i et les autres classes.
- $P[i].MAX$.index désigne l'indice de la classe la plus proche de la classe i.
- I répertorie les classes actives et non actives telles que $I[i] = 1$ si la classe i est active et 0 sinon.

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**
  fin
  $I[i] = 1$; # **Indicateur de classe active**
  construire $P[i]$, file de priorité pour d_i triée suivant $S[i][:]$;
fin
$L \leftarrow \emptyset$;
Construction du dendrogramme
pour $k \in \{1, \ldots, N-1\}$ **faire**
  $i_1 = \underset{i:I[i]=1}{\text{argmax}}\, P[i].MAX.\text{sim}$;
  $i_2 = P[i_1].MAX.\text{index}$;
  $L \leftarrow L.\text{append}([i_1, i_2])$; # **Ajout de** (i_1, i_2) **à l'ensemble des fusions**
  $I[i_2] = 0$;
  supprimer $S[i_1][i_2]$ de $P[i_1]$;
  pour *i tel que* $I[i] = 1$ *et* $i \neq i_1$ **faire**
    supprimer $S[i][i_1]$ de $P[i]$;
    supprimer $S[i_1][i]$ de $P[i_1]$;
    supprimer $S[i][i_2]$ de $P[i]$;
    $S[i][i_1] \leftarrow max\{sim\,(d_i, d_1)\,, sim\,(d_i, d_2)\}$; # **Méthode du lien unique**
    $S[i_1][i] \leftarrow max\{sim\,(d_i, d_{i_1})\,, sim\,(d_i, d_{i_2})\}$;
    insérer (avec tri) $S[i][i_1]$ dans $P[i]$ et $S[i_1][i]$ dans $P[i_1]$;
  fin
fin
Sortie : la liste des fusions L

Algorithme 1 : Algorithme de partitionnement hiérarchique agglomératif pour le cas de la méthode d'agrégation du lien unique.

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 (sur feuille libre **à rendre**) le dendrogramme associé au partitionnement hiérarchique obtenu. Vérifiez la cohérence des résultats.