# Travail pratique 3

## Clustering… et KNN

Vous avez bâti un ensemble de données impressionnant. Vous pouvez maintenant l’utiliser pour faire des expériences. La méthode que je vous demande d’explorer est le partitionnement de données à l’aide de centroïdes (Centroid-based data clustering), aussi nommé le K-means.

Vous devrez tester plusieurs configurations différentes, les principales étant la taille de fenêtre (de cooccurrences) et le nombre de centroïdes. La taille et la qualité du corpus d’entraînement pour les cooccurrences auront aussi un effet sur vos tests.

Dans un second temps, vous devez implémenter le KNN (K-nearest neighbors) à l’aide d’un corpus étiqueté qui vous sera fourni suite à l’amorce du premier objectif, soit le partitionnement de données. Le KNN servira à prédire la catégorie grammaticale « théorique » des centroïdes. Nous y reviendrons.

#### Code

Vous devez fournir tout ce qui est nécessaire afin de pouvoir reproduire vos expériences. Laissez les données que vous avez utilisées sur votre BD. Les vecteurs de cooccurrences doivent donc être calculés pour une taille de fenêtre 5, et ce AVANT la remise, c-à-d le lancement de votre code ne doit pas engendrer un réentraînement.

#### Type d’application

Cette application à la ligne de commande doit imprimer les résultats une fois les calculs terminés. Voir le fichier resultats.txt. La section à la toute fin est la plus importante. Je vous conseille toutefois d'imprimer une trace concise des états internes (itérations) de votre programme, tel qu'on peut voir au milieu du fichier.

Vous devez afficher -n (voir la section suivante) mots par cluster à la fin de l’exécution de votre programme. Triez-les par ordre de proximité du centroïde du cluster auquel le mot appartient.

#### Arguments

Vous devez gérer des options. Elles peuvent être fournies dans n’importe quel ordre.

* -c : clustering
* -t <*taille*> : taille de fenêtre. <*taille*> doit suivre -t, précédé d’un espace
* -n <*nombre*>: nombre maximal de mots à afficher par cluster (à la fin de l’exécution)
* -k <*nombre*> : nombre de centroïdes, une valeur entière.

Exemple d’appel : Y:\Cooccurrences\src>main.py -c -t 5 -k 5 -n 10

Les détails d’exécution pour le KNN vous seront fournis plus tard.

#### Init

L’option « k » détermine le nombre de vecteurs de votre matrice à choisir pour amorcer l’algorithme de partitionnement. Sélectionnez-les aléatoirement. Ces vecteurs sont les coordonnées initiales des centroïdes.

#### Tests

Ne vous gênez pas de demander beaucoup de votre ordinateur. C’est-à-dire que vous pouvez faire des tests avec 10, 20, 50, 100 centroïdes, ou plus encore. Vous verrez la nécessité d’afficher des états internes lorsque vos tests nécessiteront plusieurs heures (!) à effectuer. Le fichier resultats.txt montre la trace d’une exécution typique. Notez le temps d’exécution à la toute fin. Cette configuration ne demandait que 5 centroïdes. Vous feriez bien de rediriger vos résultats dans des fichiers à part.

Exemple de redirection :

Y:\Cooccurrences\src>main.py -c -t 5 -k 5 -n 10 > resultats\_t5\_k5\_n10.txt

Vos tests peuvent porter sur :

* La performance
  + Temps d’exécution en fonction de…
  + Nombre d’itérations en fonction de…
  + etc
    - Faites des graphiques!
* L’analyse qualitative des résultats en fonction de :
  + Taille de fenêtre
  + Type et qualité de corpus d’entraînement
  + Nombre de centroïdes
  + etc
    - Notez vos conclusions!

Il est important de ne pas lésiner ici, car un examen à la fin de la session confirmera votre niveau de curiosité.

#### Équipes

Gardez les mêmes équipes que pour le TP2.

#### À remettre

Dans le même dépôt git que pour les TPs 1 et 2, créez un répertoire TP3 et mettez-y les fichiers source Python, le fichier BD et un README spécifiant les configs qui fonctionnent (et vos noms, bien sûr). N’oubliez pas que les tests se feront à la ligne de commande.