**Clustering - Rapport v1**

**I. Import :**

**Import Packages**

Importation de tous les packages nécessaires au bon déroulement du code : pandas, numpy, matplotlib, sklearn, nltk.

**Import Data**

Fichier DataFrame:

* Data.json
* df\_final\_clean.json

**II. Functions :**

**Function dfFixer**

paramètres d’entrée:

* le chemin vers le fichier Data (.json)
* le chemin vers le fichier df\_final\_clean (.json)

Paramètre de sortie: Un DataFrame qui contient les lignes du df\_final\_clean, avec les colonnes art\_tag et art\_title du fichier Data.

Cette fonction permet de faire les prétraitements nécessaires (structurer les art\_tags) pour que la manipulation avec le DataFrame soit possible, en supprimant les doublons, les None, les chaînes vides et les “Not Found”.

**Function clusterizer**

Paramètre d’entrée : le chemin vers les données (.json).

Paramètre de sortie : le DataFrame en entrée, avec une colonne “labels” qui indique le label de chaque article

Cette fonction permet de prendre un DataFrame des articles avec des tags, et en fonction des tags et contenus des articles, regroupe les documents en 9 clusters.

**III. Méthode :**

Les données que nous avons étudiées sont composées d’articles ayant tous un contenu, pour la plupart un titre et des tags. L’objectif de ce groupe est de créer des clusters sur ces articles. Le premier problème rencontré a été que nos données ne sont pas labellisées. Nous ne pouvons donc pas faire de clustering supervisé sur ces données.

Nous avons d’abord opté pour une approche semi-supervisée. Nous avions effectué une labellisation manuelle des articles par rapport à leurs tags et nous dégageons ainsi pour un tag, un label. Chaque document avait ainsi plusieurs labels, nous comparions celui qui apparaissait le plus de fois et le choisissons comme label final. Si deux labels apparaissent le même nombre de fois, nous regardons la similarité de chaque label avec le titre de l’article. Le problème rencontré est que tous les articles n’ont pas de tags ou de titres et parfois les tags ne sont pas exploitables (abréviation, tag qui réfèrent à plusieurs labels) et nous nous sommes orientés vers une nouvelle démarche.

Ainsi, une fois les données plus propres nous avons calculé le TF-IDF des tags et nous avons entraîné et utilisé un algorithme de Kmeans afin d’obtenir des articles labellisés qui constitueront notre jeu de données d'entraînement. Puis, à partir de ces données labellisées nous avons entraîné un algorithme de Kmeans sur le TF-IDF, cette fois, du contenu des articles précédemment labellisés. Ce processus est semi-supervisé car nous avons nous même assigné des labels au jeu de données d'entraînement. Avec ce modèle nous avons cherché alors à prédire le contenu des articles n’ayant pas de tag, un thème.

**IV. Modèles / Représentations :**

1. **TF-IDF**

Méthodes de pondération servant à labelliser nos documents. Il traite les contenus des documents et évalue l’importance d’un mot dans un document. On utilise les résultats obtenus pour effectuer l'entraînement d’un modèle afin de clusteriser nos données.

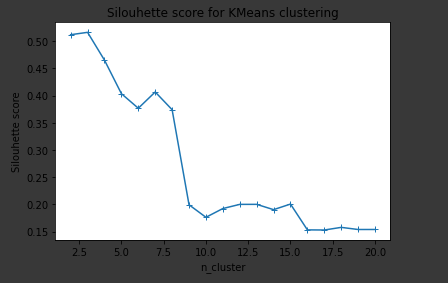
1. **k-Means**

Méthode de partitionnement des données.

Etant donné des documents, la problématique est de les regrouper en k groupes appelés clusters. On cherche à trouver le nombre optimal de clusters en utilisant la silhouette. Ce principe mesure la différence entre la distance moyenne de les points du même cluster et la distance moyenne avec les points des autres clusters.

Plus cette valeur est maximale, plus notre nombre de clusters est optimal. Pour choisir un nombre de clusters qui évolue au fil du temps, on se place en bas de la pente de la courbe et on compare chaque valeur de la silhouette.

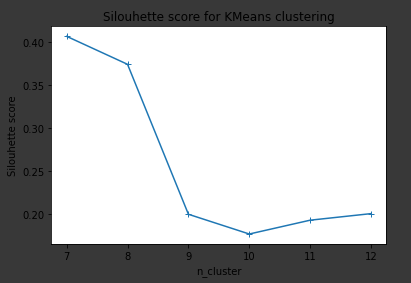
Dans un premier temps, nous avons travaillé sur un nombre de clusters allant de 0 à 20 et on visualise la figure ci-après



Sur ce graphique, on remarque que la pente se situe approximativement entre 2 et 8.

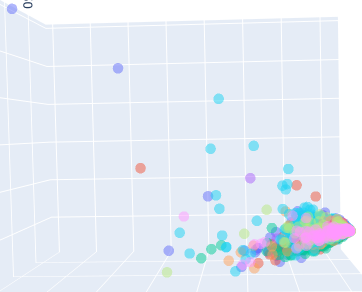
On décide alors de zoomer sur le graph en dehors de ces valeurs-là .

On reprend donc le même travail que précédemment, mais cette fois-ci nous avons travaillé sur un nombre de clusters allant de 9 à 12 , et on visualise le graphe ci-après .



On choisit finalement la valeur 9 comme nombre de clusters idéal.

Après entraînement du modèle des Kmeans sur le TF-IDF du contenu de chaque article nous obtenons la répartition des points suivant.



**V. Résultat :**

Dans chaque exécution du code, la numération des clusters différent, mais les tailles des clusters restent constants, la taille des clusters qu’on a obtenue est:

| 0 | 2758 |
| --- | --- |
| 1 | 844 |
| 2 | 479 |
| 3 | 297 |
| 4 | 639 |
| 5 | 1592 |
| 6 | 221 |
| 7 | 543 |
| 8 | 160 |

**VI. Conclusion :**

La représentation graphique ci-dessus a été réalisée sur des résultats obtenus par la méthode TF-IDF avec le modèle Kmeans, cette méthode est rapide, mais pas assez performante au vu des résultats.

En effet, choisir un nombre de clusters n’est pas forcément intuitif. Un grand nombre de clusters peut conduire à un partitionnement trop fragmenté des données.

Ce qui empêchera de découvrir des labels intéressants dans les données. Par contre, un nombre de clusters trop petit conduira à avoir, potentiellement, des clusters trop généralistes contenant beaucoup de documents.

Le tableau ci-dessus présente la répartition des documents dans chaque cluster. On remarque que les documents sont bien répartis dans chaque cluster bien que la représentation graphique montre que les nuages de points sont trop homogènes.

On a donc cherché à la demande du client, dans notre v1 à trouver automatiquement le nombre de clusters optimal ainsi qu’un meilleur modèle d’apprentissage sur des données ainsi que de nouvelles représentations.