

## Rapport TP3 :

### 1. 2. 3.

Dans cette première partie de TP, nous allons procéder à un clustering des fichiers textuels de CAMille pour la décennie 1960-1966.


Le clustering est une méthode d'apprentissage automatique qui consiste à regrouper en plusieurs clusters des points de données par similarités et par distances. Plus les points de données sont proches entre elles, plus elles présentent des propriétés similaires, et inversement. C'est une méthode d'apprentissage non supervisée et une technique populaire d'analyse statistique des données.

Pour pouvoir effectuer une analyse textuelle par clustering de nos documents, nous devons tout d'abord les transformer en données numériques manipulables. Pour cela nous allons *tokéniser*, ou autrement dit segmenter le texte de chacun de nos fichiers en *tokens*. Un *token* correspond à une chaîne de caractères entre deux espaces et traite les signes de ponctuations comme des items séparés.

Afin d'optimiser le clustering, nous allons nettoyer ces *tokens* d'éventuels *stopwords*, c'est-à-dire d'éventuels *tokens* qui apparaissent très fréquemment mais qui n'apporte pas de sens à la phrase.

Nous allons ensuite procéder à une vectorisation (= une représentation numérique) des *tokens* à l'aide de la méthode de pondération TF-IDF. C'est une méthode qui pondère au sein d'une matrice chaque *token* en fonction de son apparition au sein d'un document et du corpus. Un vecteur de *token* peut être considéré comme un point dans un espace multidimensionnel, où chaque dimension représente un aspect ou une caractéristique particulière du *token*. Le fait de représenter les *tokens* de chaque document sous forme de vecteur numérique facilite le calcul de distance entre ceux-ci.

La matrice étant générée, nous pouvons connaître ainsi la valeur des vecteurs de nos *tokens*. Voici un exemple ci-dessous des vecteurs du document n°7 (voir Capture d'écran n°1 à la page n°2) :

Ces résultats nous indiquent une similarité évidente entre les mots « entretien », « entrée », « env », « environ », et «  » avec une même valeur de vecteur à 0.000000.

Les chiffres « 58 » et « 57 » se rapprochent fortement  $\approx 0.34$ , alors que le chiffre 56 lui est juste en dessous à une valeur de 0.212987.

Puis les mots « voitures » et « luxe » n'ont pas beaucoup de distances entre eux.

Nous pouvons conclure avec cet exemple que les distances entre les *tokens* semblent relativement justes, seule la valeur du caractère « ♦ » semble anormale. Pour optimiser les distances entre les *tokens* de notre corpus nous pourrions encore mieux nettoyer nos documents en enlevant les chiffres et les signes de ponctuation par exemple. Cela permettrait d'appliquer l'analyse non plus sur des données textuelles brutes, mais sur une liste comportant seulement des mots.

```
pd.Series(  
    tfidf_vectors[7].toarray()[0],  
    index=vectorizer.get_feature_names_out()  
).sort_values(ascending=False)  
[452] ✓ 0.1s
```

...	58	0.345516
	57	0.340743
	56	0.212987
	voitures	0.181692
	luxé	0.176218
	...	
	entretien	0.000000
	entréé	0.000000
	env	0.000000
	environ	0.000000
	♦	0.000000
	Length: 1098, dtype: float64	

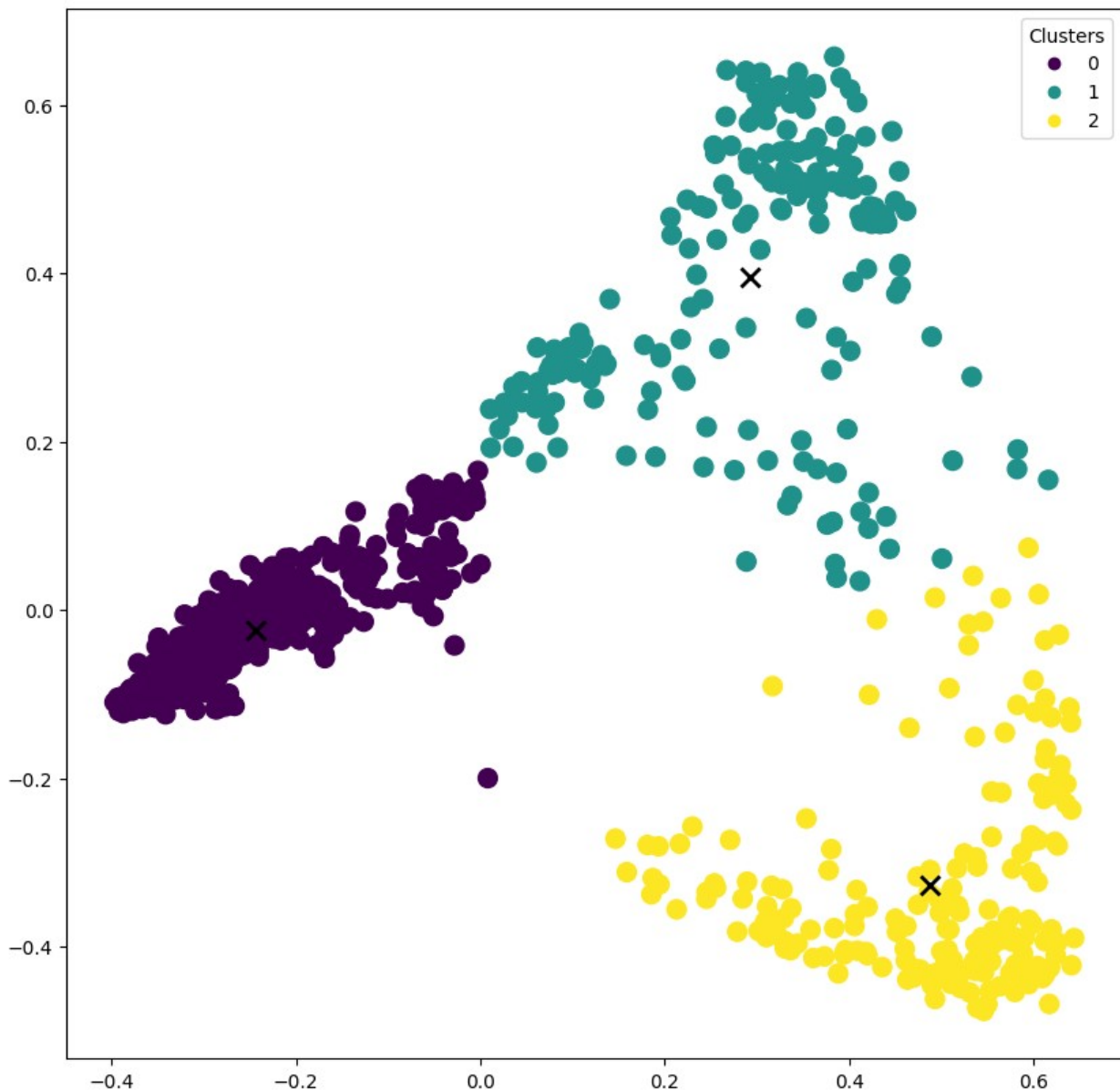
Capture d'écran n°1

Nous allons à présent appliquer un nombre de cluster.

Voici ci-dessous (voir Capture d'écran n°2 à la page n°3) la visualisation en 2 dimensions pour un nombre de 3 clusters de notre corpus :

Les croix noires sont les centre des clusters. Elles correspondent aux zones de densité relativement élevées, c'est-à-dire les zones où beaucoup de points de données sont proches par rapport à d'autres zones.

Les clusters sont relativement distincts les uns des autres. Cela peut laisser songer qu'il existerait trois grand-es thématiques, sujets, termes qui ont été répété-es régulièrement sur la décennie 1960-1966 dans le corpus de CAMille. Mais nous ne pouvons pas avec ces résultats faire de conclusion objective puisqu'il nous faudrait analyser en détail les *tokens* un par un.



Capture d'écran n°2

#### 4. 5. 6.

Dans la seconde partie du TP, nous allons entraîner un modèle word2vec (word embeddings) sur des phrases prêt segmentées à partir du fichier sens.txt.

Word2vec a été développé par une équipe de recherche chez Google sous la direction de Tomas Mikolov dans les années 2010. Il s'agit d'un ensemble de méthode d'apprentissage visant à représenter les mots d'un texte par des vecteurs de nombres réels. Il repose sur des réseaux de neurones artificiels à deux couches entraînés pour reconstruire le contexte linguistique des mots. De cette manière, les mots qui partagent

des contextes similaires sont représentés par des vecteurs numériques proches et occupent des positions spatiales proches.

Les mots choisis pour expérimenter les différentes fonctions qui suivent ont été choisies de manière intentionnelle pour questionner les textes selon certains stéréotypes liés à la couleur de cheveux.

Ci-dessous (voir Capture d'écran n°3) nous avons calculé avec la fonction `similarity` la similarité entre deux mots. Plus le nombre se rapproche de 1, plus il y a de similitudes entre les mots.

```
Calculer la similarité entre deux termes

[30] ✓ 0.0s
... 0.8103219
model.wv.similarity("belle", "ravissante")

[33] ✓ 0.0s
... 0.47960347
model.wv.similarity("femme", "ravissante")

[40] ✓ 0.0s
... 0.14339742
model.wv.similarity("homme", "ravissant")

[35] ✓ 0.0s
... 0.8873249
model.wv.similarity("blonde", "ravissante")

[37] ✓ 0.0s
... 0.8173834
model.wv.similarity("rousse", "ravissante")

[38] ✓ 0.0s
... 0.6901406
model.wv.similarity("brune", "ravissante")

[39] ✓ 0.0s
... 0.7705526
model.wv.similarity("ravissante", "naïve")

[40] ✓ 0.0s
... 0.71863866
model.wv.similarity("blonde", "naïve")

[50] ✓ 0.0s
... 0.76688427
model.wv.similarity("blond", "naïf")

[41] ✓ 0.0s
... 0.35653964
model.wv.similarity("brune", "naïve")

[51] ✓ 0.0s
... 0.43951243
model.wv.similarity("brun", "naïf")

[42] ✓ 0.0s
... 0.6058221
model.wv.similarity("rousse", "naïve")

[52] ✓ 0.0s
... 0.25419548
model.wv.similarity("roux", "naïf")
```

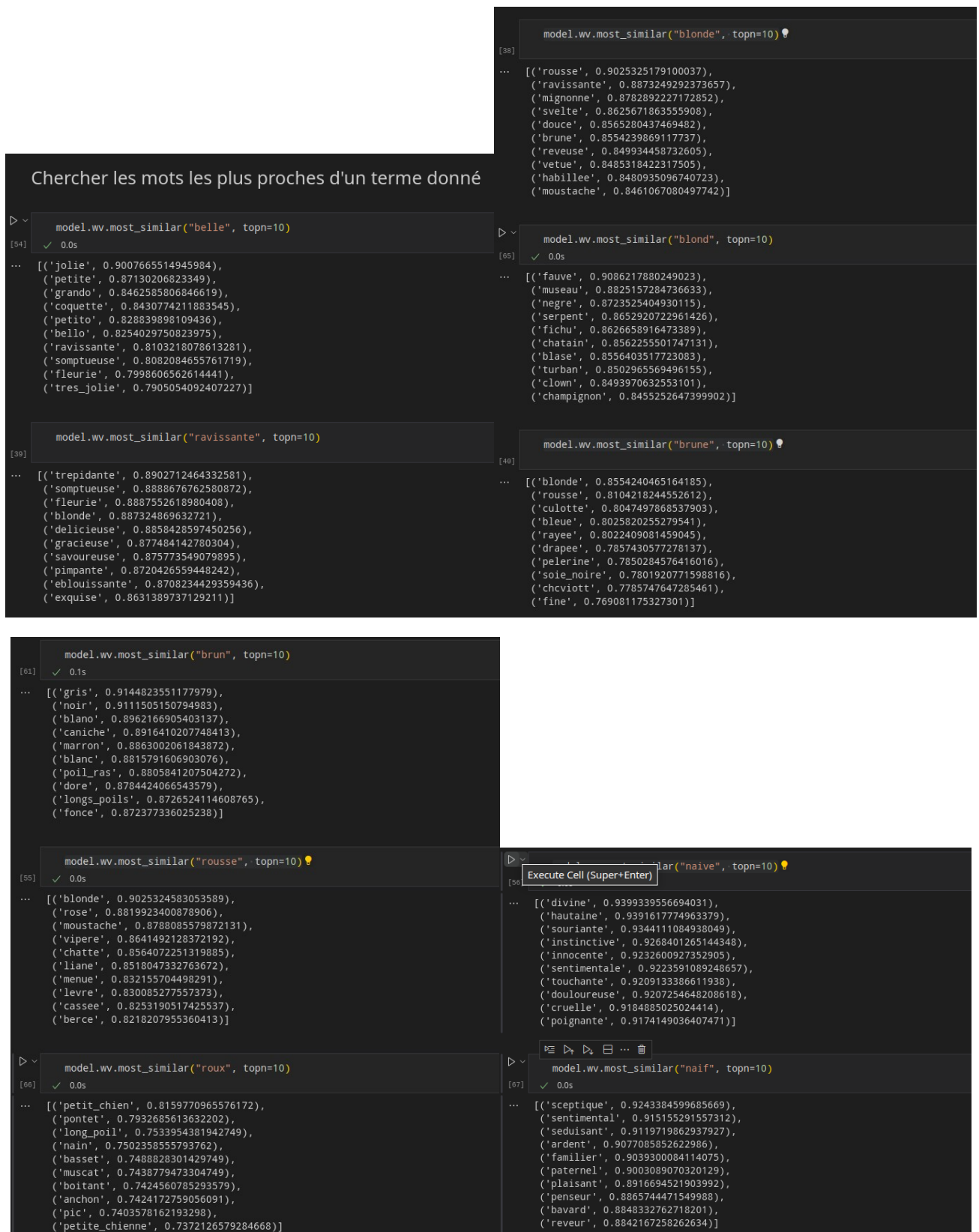
Capture d'écran n°3

Le premier test de similitude semble correct, effectivement les mots « belle » et « ravissante » sont proches dans le lexique francophone.

Nous avons ensuite fait différents tests qui semblent affirmer des stéréotypes populaires : une personne de couleur de cheveux blonde serait plus ravissante qu'une personne avec des cheveux bruns, ou encore qu'une personne ayant des cheveux blonds serait plus naïve qu'une personne de couleur de cheveux brune.

Nous pouvons observer que si nous mettons les noms qui qualifient la couleur de cheveux (blonde, brune, rousse) au masculin (blond, brun, roux) en lien avec le mot « naïf » les résultats de similitudes sont plus élevés. Une potentielle explication serait du au nombre plus important de mots masculins que féminins dans la langue française.

Nous allons tester maintenant la fonction `most_similar` avec ces différents mots utilisés dans la fonction précédente (voir Capture d'écran n°4) :



Capture d'écran n°4

Les tests semblent juste lexicalement.

Nous remarquerons que dans les mots les plus similaires aux noms « blonde », « brune » et « rousse », nous retrouvons des mots connotés au genre sexuel féminin comme :

```
('chatte', 0.8564072251319885),  
( 'levre', 0.830085277557373),  
( 'culotte', 0.8047497868537903).
```

Alors que pour les mêmes noms au masculin nous retrouvons plutôt des termes liés à l'animal du chien :

```
('long_poil', 0.7533954381942749),  
( 'petite_chienne', 0.7372126579284668),  
( 'poil_ras', 0.8805841207504272),  
( 'longs_poils', 0.8726524114608765),  
( 'caniche', 0.8916410207748413),  
( 'museau', 0.8825157284736633).
```