# Exemple d'image

# 

# 

# PPDligne horizontale

## Fouille de texte pour l'exploration des facteurs de sévérité du COVID-19

### Proposé par :

## Séverine Affeldt & Lazhar Labiod

### Réalisé par :

## Mohammed Erifai MAAMIR - MLDS

## Kamel MESBAHI -MLDS

## Ryad lotfi MAHTAL - MLDS

## Ahmed Seyfeddine GOUMEIDA - MLDS

# 

# Table des matières :

[**Table des matières :**](#_bszvemiewsy) **2**

[**Introduction**](#_uxtibeg4x6kw) **3**

[**Chargement de données**](#_uxi4h37gxuh1) **4**

[**Prétraitement de données**](#_t1q8ypmsv226) **5**

[Suppression des valeurs null](#_22tz1u5fjt96) 5

[Multi-langues traitement](#_20i5sjokbk0) 5

[Stopwords](#_v5s0my7f2qmr) 6

[Ponctuations](#_b01ol947fznk) 6

[Lowercase](#_se871ap0uu6p) 7

[Lemmatization](#_srigbfs2fjuh) 7

[Liens et HTML tags](#_yo4wht5uh66g) 7

[Frequent words](#_lqiwr32ok6t1) 8

[Rare words](#_acri5qrdi8b6) 8

[Nombre de ‘mots’ / ‘mots uniques’ par articles](#_587dm1w0kgfl) 8

[**Vectorization**](#_ng5sg5pq0wg7) **9**

[**CoClustering & Topic modeling:**](#_8n2yi8yfy3zp) **10**

[CoClustering:](#_ry075o4e0llk) 10

[Data Collection :](#_9mmpkzpg1t5e) 11

[Number de Co-Clusters :](#_1sc9vhvqtmj8) 13

[Visualisation de la matrice Cocluster :](#_af4hvx6n5grm) 14

[Top Terms :](#_ao57avfvu4zv) 15

[Taille de chaque cluster :](#_f8m2qlub8rx) 16

[Similarité entre les mots du même cluster:](#_uab2zsdlb2ka) 17

[**Named Entity Recognition (NER):**](#_hpc5qrs2gkbn) **18**

[**Conclusion**](#_ut64c1e8aq3z) **19**

[**Références / Sources**](#_u4qu9havepku) **20**

# Introduction

Le text mining, également appelé traitement automatique du langage, peut être défini comme étant un ensemble de techniques issues de l’intelligence artificielle, alliant plusieurs domaines : la linguistique, la sémantique, le langage, les statistiques et l’informatique. Combinées ensemble, ces techniques permettent d’extraire des données pour recréer de l’information à partir de corpus de textes en les classifiant et les analysant de manière à établir des tendances. Le text mining est notamment employé dans le secteur du marketing, mais également dans de nombreux autres domaines tels que la communication, les sciences politiques, la recherche et la médecine.

L’intelligence artificielle est désormais capable de classifier automatiquement les textes par sentiment, par sujet ou par intention. Un algorithme de Text Mining est par exemple capable de passer en revue les commentaires sur un produit pour déterminer s’ils sont principalement positifs, neutres ou négatifs. Il est aussi possible de repérer les mots-clés les plus fréquemment employés. Même s’il est loin d’être un concept nouveau, le text mining connaît aujourd’hui un nouvel essor. L’émergence du Big Data où le stockage des données numériques n’est plus un problème et où les sources de données disponibles se multiplient font de l’analyse des données textuelles un enjeu crucial.

Alors même que des études scientifiques sont publiées chaque jour et que de nouveaux projets de recherche sont lancés dans le monde entier, nous avons tous plus de questions que de réponses sur le nouveau coronavirus et le COVID-19. Et les questions s'accumulent. Historiquement, ce type de questions trouve des réponses au fil du temps grâce à des recherches approfondies qui sont publiées, examinées et utilisées pour aider à prévenir et à traiter les itérations futures d'une maladie infectieuse.

Avec le COVID-19, nous n'avons pas le luxe du temps. Bien qu'il y ait beaucoup de recherches et de données partagées sur la maladie, le temps nécessaire pour lire, comparer et comprendre toutes les recherches tout en combattant la maladie est un problème croissant. Le problème est le suivant : quelle stratégie analytique pouvons-nous employer pour mettre en relation les bonnes recherches et les bonnes personnes afin de répondre à certaines de nos questions les plus urgentes ? et là l’exploitation de la littérature scientifique s'introduit pour explorer certains proﬁls de patients afin d’identiﬁer les comorbidités et les facteurs de sévérité du COVID-19 afin de proposer des traitements innovants susceptibles de réduire la sévérité de la maladie.

# 

# Chargement de données

Pour la récupération des données on a choisi de le faire d’une manière dynamique qui va nous permettre de jouer sur le nombre d’articles voulus, les termes utilisés pour chercher les articles et même la mise à jour de données.

Pour ce faire, on a utilisé le module Entrez du package Bio, ce module permet d'accéder aux bases du NCBI (à la fois les bases type séquences et autre, mais aussi PubMed par exemple).

La plupart des requêtes récupère les infos sous forme de XML (Nous on a choisi le format json) et il y a un parseur intégré que l'on peut appeler avec Entrez.read qui transforme le xml en structure python (avec dictionnaires et listes) puis on met les données dans un Dataframe pour pouvoir les exploiter dans les prochaines étapes.

Pour la recherche on a choisi de faire la requête avec les termes “Covid”,”Coronavirus”,”Covid19”,”Severety” avec un nombre d’articles égale à 5 000 et en prenant que les abstracts des articles, on a rapidement constaté que 10 000 enregistrement c’est très peu pour notre étude donc pour récupérer plus de 10 000 UID, on a soumis plusieurs demandes de recherche tout en augmentant la valeur de retstart (retstart est le paramètre qui définit l’index séquentiel du premier UID dans le jeu récupéré à afficher dans la sortie de la requête effectuée).

Le module Entrez enregistre une adresse mail et un nom d'outil de façon à pouvoir nous contacter en cas de problème (option facultative).

max\_searchs = 5000

terms = "coronavirus covid covid19 severity"

Entrez.email = 'A.N.Other@example.com'

ids=[[]]

print("max\_searchs : ",max\_searchs)

for i in range(0,max\_searchs,100):

print(i," ", end='')

h = Entrez.esearch(db='pubmed', retmax=100,retstart=i, term=term)

result = Entrez.read(h)

ids.append(result['IdList'])

h = Entrez.efetch(db='pubmed', id=ids, rettype='medline', retmode='json')

records = Medline.parse(h)

# Prétraitement de données

Le Data Cleaning (nettoyage de données) est l’étape la plus importante avant d’analyser ou modéliser des données, maintenant que nous avons chargé notre jeu de données, nous devons nettoyer le texte pour améliorer les performances de classification ou du clustering. Dans ce qui suit, nous allons parcourir un certain nombre de tâches de Data Cleaning.

## Suppression des valeurs null

Dans le monde réel, les données sont rarement propres et homogènes. Les données peuvent être manquantes pendant l'extraction ou la collecte des données. Les valeurs manquantes doivent être traitées car elles réduisent la qualité de nos mesures de performance. Elles peuvent également conduire à une prédiction ou à une classification erronée et causer un biais élevé pour tout modèle utilisé.

Pour se débarrasser des valeurs null on a utilisé la fonction dropna de Pandas qui supprime chaque ligne contenant une valeur nulle. Bien évidemment qu’il existe plusieurs méthodes pour traiter les valeurs nulls, notre choix de supprimer la ligne qui contient au moins une valeur null est le plus adéquat par rapport au type de nos données , donc on peut pas se servir des autres méthodes utilisées dans le cas numérique (par exemple : remplacement par la valeur moyenne de la colonne ).

df.dropna(inplace=True)

## Multi-langues traitement

Puisque on veut travailler qu’avec des articles en anglais, nous allons déterminer la langue de chaque article dans le cadre de données. Toutes les sources ne sont pas en anglais et la langue doit être identifiée pour que nous sachions quoi garder et quoi supprimer.

Pour ce faire, on a utilisé le package “langdetect” qui supporte 55 langues dont pour chaque article on a détecté la langue et on l’a mémorisé dans une liste qu’on va l'insérer après comme une nouvelle colonne ”language” dans notre Dataframe. Cette démarche va nous permettre de connaître les articles à garder (ceux qui ont la valeur “en” dans le champ language).

df['language'] = languagesdf = df[df['language'] == 'en']

## 

## 

## Stopwords

Une partie du prétraitement consistera à trouver et à supprimer les stopwords (mots qui serviront de bruit dans l'étape de clustering et topic modeling). Pour cela on s’est servi de la liste “STOP\_WORDS” du package spacy qui contient les stopwords qui sont utilisés dans n’importe quel texte d’anglais, par contre les documents de recherche utilisent souvent des mots qui ne contribuent pas réellement au sens et qui ne sont pas considérés comme stopwords dans la liste “STOP\_WORDS” qu’on vient de citer, donc on a décidé de rajouter à cette liste une deuxième liste qui contient les stopwords généralement contenus dans les documents de recherche. On a créé une fonction qu’on a appelé “spacy\_process()” qui va enlever tous les mots qui appartiennent à la liste des stopwords.

from spacy.lang.en.stop\_words import STOP\_WORDS

stopwords = list(STOP\_WORDS)

for word in mytokens if word.text not in stopwords

## Ponctuations

Dans le même contexte et pour faire plusieurs traitement sur le texte, on a utilisé le package “en\_core\_web\_lg” (English pipeline optimized for CPU, qui contient les composantes: tok2vec, tagger, parser, senter, ner, attribute\_ruler, lemmatizer) pour enlever les ponctuations de notre texte en utilisant la liste punctuations du package “string” dont pour chaque mot on va tester s’il est dans la liste “punctuation”, si oui on le supprime.

import string

punctuations = string.punctuation

for word in mytokens if word.pos\_ != "PUNCT"

## Lowercase

Toujours dans la fonction “spacy\_process()” et en utilisant la fonction prédéfinie “lower()” du package “en\_core\_web\_lg”, on a transformé tous nos mots en minuscule (lowercase).

mytokens2 = [word.lemma\_**.lower()**.strip()

## Lemmatization

Le processus de lemmatisation consiste à représenter les mots (ou lemmes ) sous leur forme canonique. Par exemple pour un verbe, ce sera son infinitif. Pour un nom, son masculin singulier. L'idée étant encore une fois de ne conserver que le sens des mots utilisés dans le corpus.

Dans notre démarche, on a exécuté la lemmatisation en deux étapes : d’abord en utilisant le lemmatizer du package “en\_core\_web\_lg” qu’on a appelé en exécutant la fonction “spacy\_process()”.

nlp = en\_core\_web\_lg.load()

mytokens = nlp(phrase)

mytokens2 = [word**.lemma\_**.lower().strip() for word in mytokens]

Ensuite on l'a fait une deuxième fois avec “WordNetLemmatizer” du package “nltk.stem”.

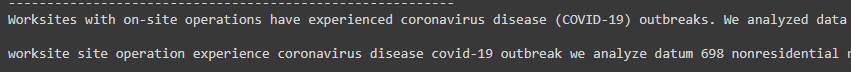
import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

**lemmatizer.lemmatize(word)**

results :



## Liens et HTML tags

Lorsqu'il s'agit d'analyser du HTML, on ne souhaite probablement pas traiter de JavaScript ou de CSS intégrés, et on n'est intéressé que par le texte, l'idée est donc de construire une expression régulière qui peut trouver tous les caractères "< >" comme première incidence dans un texte, et ensuite, en utilisant la fonction sub, nous pouvons remplacer tout le texte entre ces symboles par une chaîne vide, c’est exactement ce qu’elle va faire la fonction “remove\_html(text)” qu’on a définit.

Même principe avec Html tags et en utilisant la fonction “remove\_urls(text)” qui, à l’aide d’une expression régulière (package re), trouve et supprime les liens.

def remove\_urls(text):

url\_pattern = re.compile(r'https?://\S+|www\.\S+')

return url\_pattern.sub(r'', text)

## Frequent words

Cette étape consiste à supprimer les mots fréquents dans le corpus donné. Pour ce faire, on calcule le nombre d'occurrences de chaque mot puis on supprime les mots les plus fréquents. Finalement on a décidé de ne pas enlever les frequent words car on a constaté qu’il existe dans les résultats plusieurs mots qui peuvent être intéressant pour nous dans les prochaines étapes de clustering et topic modeling.

df[".."] = df[".."].apply(lambda text: remove\_freqwords(text))

## Rare words

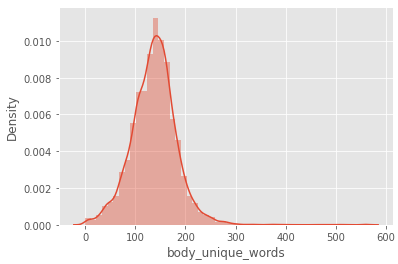
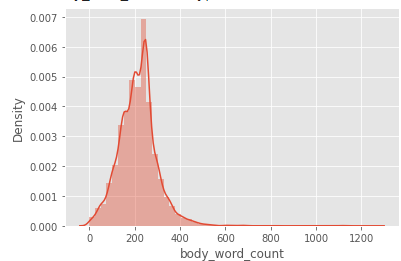
Cette étape est très similaire à l'étape précédente, mais nous allons supprimer les mots rares du corpus au lieu des mots les plus fréquents.

df[".."] = df[".."].apply(lambda text: remove\_rarewords(text))

## Nombre de ‘mots’ / ‘mots uniques’ par articles

Ces deux graphes nous donnent une bonne idée du contenu auquel nous avons affaire. La plupart des articles ont une longueur de 150 à 250 mots. Les longues queues dans les deux graphes sont dues à des valeurs aberrantes. En fait, 98% des articles font entre 150 et 250 mots, tandis qu'un petit nombre d'entre eux font plus de 250 ou moins de 150 mots.

Le nombre de mots moyen est environ 200 mots par article par contre seulement 130 mots unique par article.

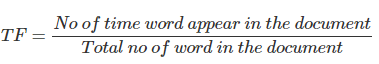


# Vectorization

Maintenant que nous avons prétraité les données, il est temps de les convertir dans un format qui peut être traité par nos algorithmes. Dans l'apprentissage automatique, les features sont essentiellement des attributs numériques à partir desquels on peut effectuer des opérations mathématiques telles que la factorisation matricielle, le produit scalaire, etc. Mais il existe plusieurs scénarios dans lesquels les ensembles de données ne contiennent pas d'attributs numériques, par exemple l'analyse sentimentale d'un utilisateur de Twitter/Facebook. Dans notre cas, le jeu de données contient des chaînes de caractères, dont la conversion de ce type de features en features numériques est appelée featurisation. Plus clairement, le processus de conversion du texte en vecteur est donc appelé vectorisation.

À cette fin, nous utiliserons tf-idf. Nous allons convertir nos données de format chaîne de caractères en une mesure de l'importance de chaque mot pour l'instance dans l'ensemble de la littérature. TF-IDF est l'abréviation de Term Frequency-Inverse Document Frequency, qui indique l'importance d'un mot dans un corpus ou un ensemble de données. TF-IDF contient deux concepts : la fréquence des termes (TF) et la fréquence inverse des documents (IDF).

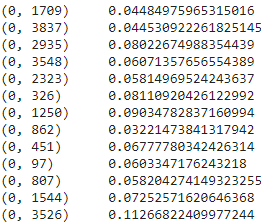
Term Frequency (TF) : est définie comme la fréquence à laquelle le mot apparaît dans le document ou le corpus. Comme chaque phrase n'a pas la même longueur, il est possible qu'un mot apparaisse plus souvent dans une longue phrase que dans une phrase plus courte. La fréquence des termes peut être définie comme suit :



Inverse Document Frequency(IDF) : La fréquence inverse des documents est un autre concept utilisé pour déterminer l'importance d'un mot. Il est basé sur le fait que les mots moins fréquents sont plus informatifs et plus importants. L'IDF est représentée par la formule :



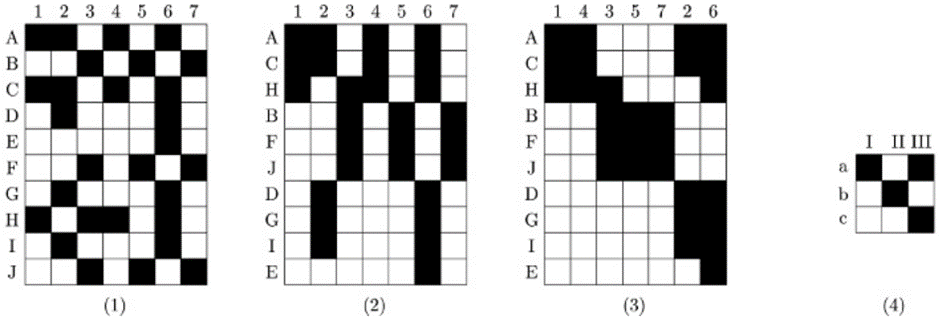
une partie d’affichage du résultat TF-IDF :



# CoClustering & Topic modeling:

## CoClustering:

La classification double ou « Biclustering » est une technique d'[exploration de données](https://fr.wikipedia.org/wiki/Exploration_de_donn%C3%A9es) non supervisée permettant de segmenter simultanément les lignes et les colonnes d'une matrice. Cette approche a été utilisé massivement en biologie - par exemple dans l'analyse de l'[expression génétique](https://fr.wikipedia.org/wiki/Expression_g%C3%A9n%C3%A9tique), mais aussi dans d'autres domaines tels que la compression d'image de synthèse, l'analyse médicale, la caractérisation d'émetteurs de [pourriels](https://fr.wikipedia.org/wiki/Spam) (« spam »), l'analyse du mouvement, l'analyse des termes publicitaires sur internet.



Figure

* a) Initial binary data matrix.
* b) Data matrix reorganized according to a partition of rows.
* c) Data matrix reorganized according to partitions of rows and columns.
* d) Summary of this matrix

## Data Collection :

Comme expliqué dans la partie “Chargement de données” La collecte des données a été faite en utilisant une Api python sur la plateforme “PubMed” regroupant des articles médicaux, afin de récupérer des données parlant d’une thématique particulière on utilise la recherche par mot clé. Dans notre cas on cherche des articles parlant du virus Covid 19, ce qui nous intéresse dans ce projet c’est de trouver les maladies qui ont une relation avec ce virus et qui peuvent éventuellement causer sa gravité chez certains malades.

### Approche 1 :

une première approche consiste à rechercher des articles par le mot clé “covid” ou “coronavirus” , cette recherche retourne un grand nombre d’articles et le filtrage a été fait d’une manière arbitraire, l’analyse des résultats des méthodes de clustering et de Co-clustering ne sont pas prometteurs et le clusters résultants ne sont pas forcément caractérisés par les noms des maladies ni par les symptômes qui leurs sont associés , nous avons quand même réussi à identifier certains clusters parlant des maladies psychologiques, des symptômes de la maladie , … mais nous n’avons pas un moyen de mesurer la proximité entre les maladies et d’identifier les symptômes communs entre elles .

Approche 2 :

une 2ème approche consiste à rechercher tous les articles qui parlent du coronavirus et d’une autre maladie , un premier test a été fait avec les mots clés “covid” et “obésité” , les clusters résultant semblent avoir plus de sens et certaines conclusions ont été faites dans les chapitres suivants sur chaque étude de cas, l'inconvénient de cette approches est le fait que les maladies qui ont un rapport avec le covid ne sont pas toutes connues et l’objectif de notre étude justement est d’identifier ces maladies d’une manière automatique sans faire appel à un expert métier .

Approche 3 :

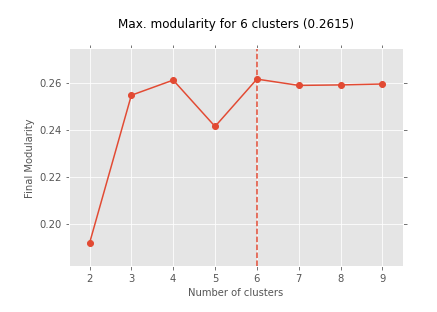
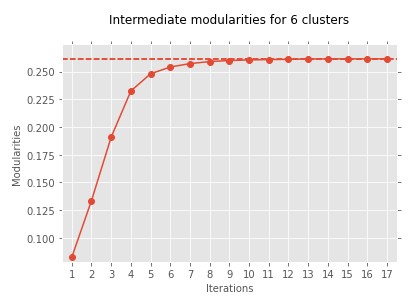
Dans la 3ème approche, nous avons utilisé les entités nommées pour récupérer une liste des maladies chroniques les plus connues, une liste des symptômes et nous les avons injectées séquentiellement dans la requête de recherche sur l’API PubMed accompagnés par les mots clés “covid” et “sévérity”

## Number de CoClusters :

Pour définir le nombre idéal des co clusters on a fait appelle à la fonction prédéfinie "best\_modularity\_partition()" on lui donnant en paramètre min\_cluster\_nbr et max\_cluster\_nbr.

Ce qui concerne la visualisation on a utilisé deux fonctions :

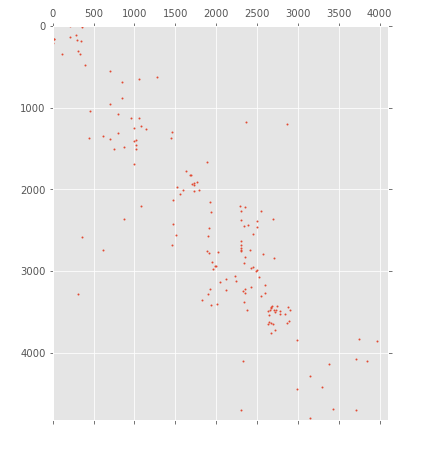
**1- “plot\_max\_modularities()” :** qui prend en paramètre le résultat de la fonction best\_modularity\_partition pour déterminer le nombre de cocluster qui a le maximum des modalités,

**2- “plot\_intermediate\_modularities()” :**  qui prend en paramètre le model récupéré de la fonction best\_modularity\_partition pour visualiser intermédiate modalities pour le nombre de cocluters déterminé dans la phase précédente. 

## Visualisation de la matrice Cocluster :

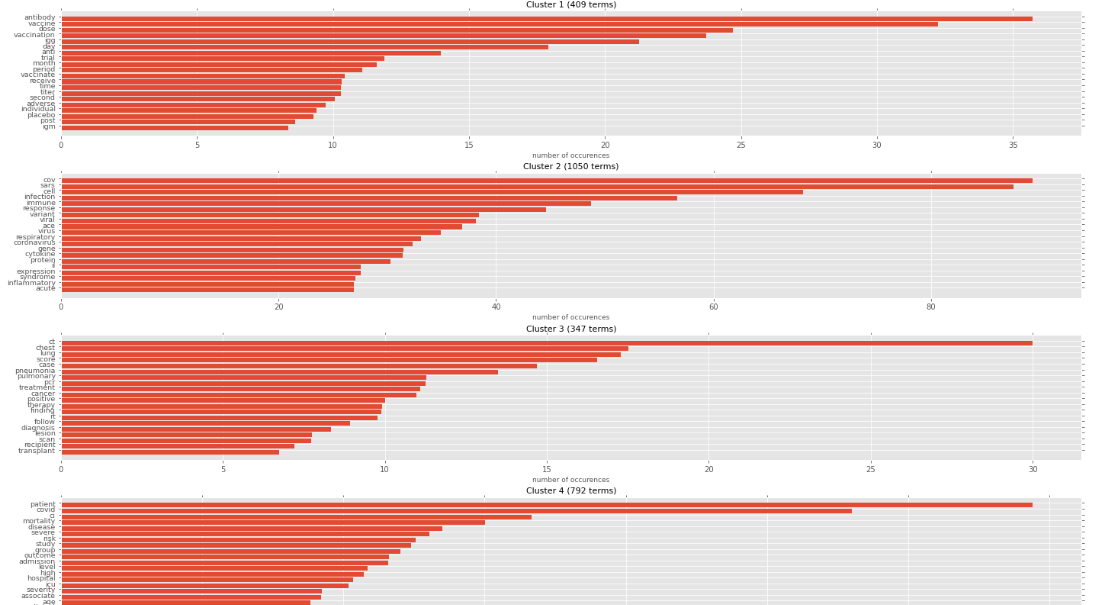
Après avoir entraîné le model , il suff.it d'exécuter la fonction plot\_reorganized\_matrix qui nous montre la matrice de Coclustering qui est une matrice carrée.

EXPLICATION DE LA MATRICE !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!



## Top Terms :

On peut facilement voir la différence entre les cluster , prennent en exemple le premier cluster qui parle beaucoup plus sur le vaccin, anti corps et les périodes ce qu’on peut lui donner le titre “Vaccin”, le troisième cluster qui contient des mots de maladies comme cancer, pneumonia, pulmonary and chest ce qui fait référence au maladies et facteurs qui peuvent jouer un role dans la sévérité et la comorbidité.

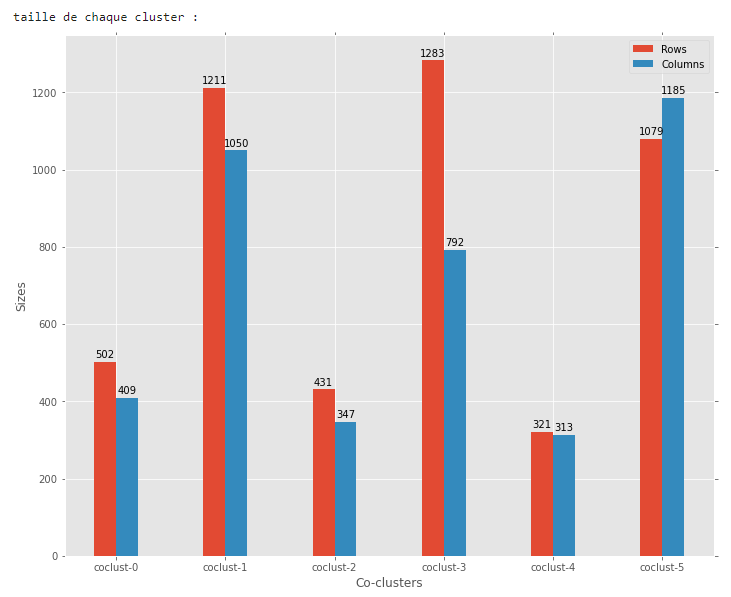


## Taille de chaque cluster :

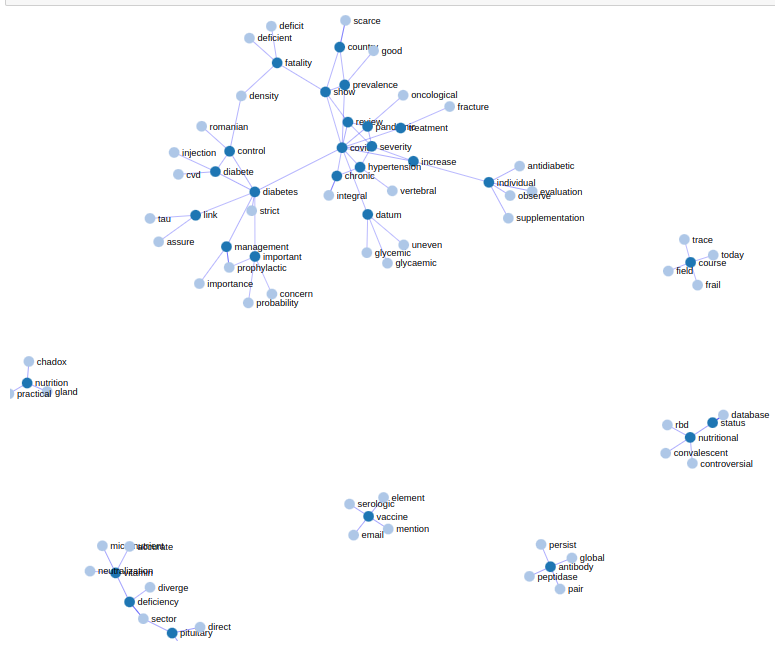
en regardant le plot on constate qu’il y’a des grands écarts entre les taille des clusters, les tailles varient entre :

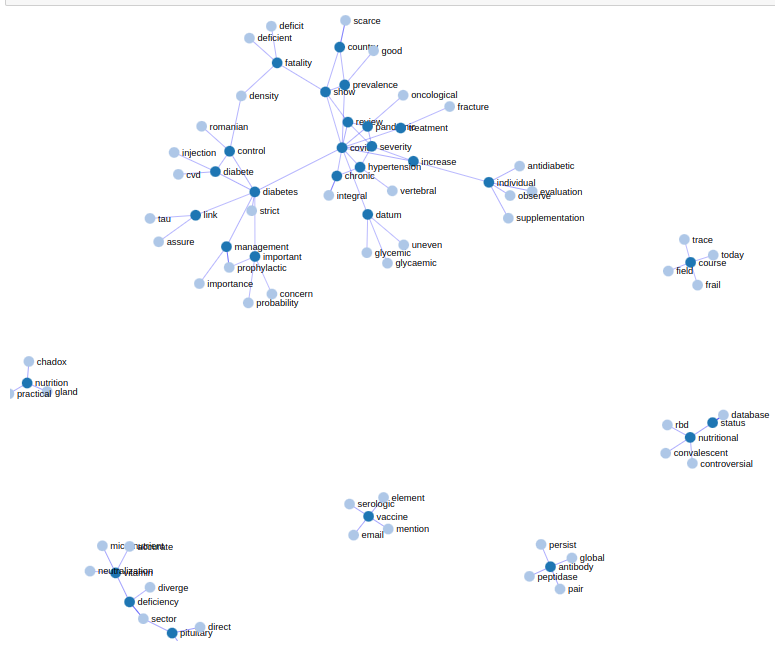
* [313 - 1185] pour les colonnes
* [321 - 1283] pour les lignes

ce qui nous pousse à vérifier si deux clusters de petites tailles peuvent parler du même sujets, dans ce cas on peut fusionner les petits cluster ou de deviser les grands.

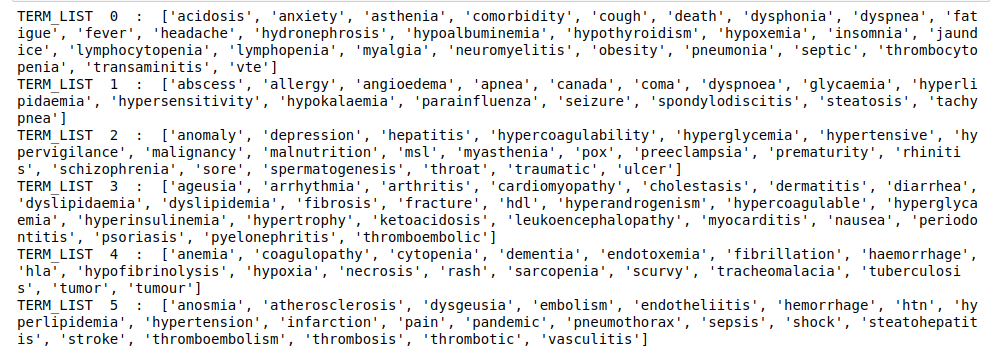


## Similarité entre les mots du même cluster:





# Named Entity Recognition (NER):

Pour aller plus loin dans l'exploitation des clusters et avec l’utilisation de l’approche NER (Named Entity Recognition), il est possible d’extraire les entités de différentes catégories. Il existe plusieurs modèles de base, pré-entraînés, comme **“en\_ner\_bc5cdr\_md"**, qu’on va utiliser et qui est capable de reconnaître les personnes, les lieux, les dates et même les maladies. vu qu’on est intéressé que par les maladies on va nous limiter par chercher les mots qui ont ***label = DISEASE***

les entitées nommé labellisées avec le mot “DISEASE” dans chaque cluster:

* cluster 0 : 27 diseases
* cluster 1 : 16 diseases
* cluster 2 : 21 diseases
* cluster 3 : 25 diseases
* cluster 4 : 18 diseases
* cluster 5 : 21 diseases

***Interprétation des résultats :***

# Conclusion

Pour développer le NLP biomédical COVID-19, des scientifiques des données, des ingénieurs logiciels, des cliniciens et des chercheurs en médecine sont réunis afin de permettre une approche éclairée et de développer un processus d'extraction des connaissances bien équilibré. L'effort multidisciplinaire émule le raisonnement inductif clinique du monde réel qui utilise une approche par étapes pour évaluer, extraire et hiérarchiser les connaissances afin de permettre une médecine fondée sur les preuves.

Alors que notre NLP biomédical extrait les entités pertinentes de la littérature biomédicale et fournit un haut degré de cohérence des clusters, il existe des limitations associées à la bibliothèque de reconnaissance des entités nommées biomédicales. Cela restreint la capacité d'extraire toutes les entités biomédicales alignées nécessaires(ça se voit clairement dans nos résultats). La reconnaissance des concepts biomédicaux est un domaine de recherche actif, et des méthodes améliorées ciblant un large éventail de types d'entités et de concepts peuvent être substituées.

La pandémie de COVID-19 a créé une situation unique où il est nécessaire d'accéder rapidement à des connaissances cliniques en constante évolution à partir des publications de qualité variable qui se multiplient de manière exponentielle. Dans ce projet on a pu exploiter la littérature scientifique disponible en ligne à l’aide d’approches de fouilles de texte avancés qui nous a également permis d’explorer les relations entre les maladies, l’étape prochaine dans cette recherche sera d'exploiter le nouveau corpus créé afin de faire la relation entre le covid-19 et des nouveaux traitements efficaces.

# Références / Sources

* [towardsdatascience.com](https://towardsdatascience.com/)
* [stackoverflow.com](https://stackoverflow.com/)
* [github.com](https://github.com/)
* [ia-data-analytics.fr](https://ia-data-analytics.fr/)
* [datascientest.com](https://datascientest.com/)
* [blogdigital.beijaflore.com/text-mining/](https://blogdigital.beijaflore.com/text-mining/)
* [fr.coursera.org/learn/python-text-mining](https://www.coursera.org/learn/python-text-mining)
* [www.udemy.com/course/project-based-text-mining-in-python/](http://www.udemy.com/course/project-based-text-mining-in-python/)
* <https://datascientest.com/text-mining-definition>
* <https://ia-data-analytics.fr/logiciel-data-mining/text-mining/text-mining-traiter-vos-donnees-textuelles/>
* <https://www.sas.com/en_us/insights/articles/analytics/covid-19-research-with-text-analytics.html>
* <https://www.researchgate.net/>
* <https://intellica-ai.medium.com/comparison-of-different-word-embeddings-on-text-similarity-a-use-case-in-nlp-e83e08469c1c>
* <https://medium.com/apprentice-journal/pca-application-in-machine-learning-4827c07a61db>
* <https://mrmint.fr/algorithme-k-means>
* <https://towardsdatascience.com/latent-semantic-analysis-deduce-the-hidden-topic-from-the-document-f360e8c0614b>
* <https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24>
* <https://medium.com/voice-tech-podcast/topic-modelling-using-nmf-2f510d962b6e>
* <https://medium.com/@paritosh_30025/natural-language-processing-text-data-vectorization-af2520529cf7>
* <https://moncoachdata.com/blog/nettoyage-de-donnees-python/>
* <http://www.python-simple.com/python-biopython/utilisation-entrez.php>
* <https://medium.com/@maheshdmahi/scispacy-for-bio-medical-named-entity-recognition-ner-63ed548f1df0>
* <https://www.datacamp.com/community/tutorials/stemming-lemmatization-python>
* <https://pubmed.ncbi.nlm.nih.gov/>