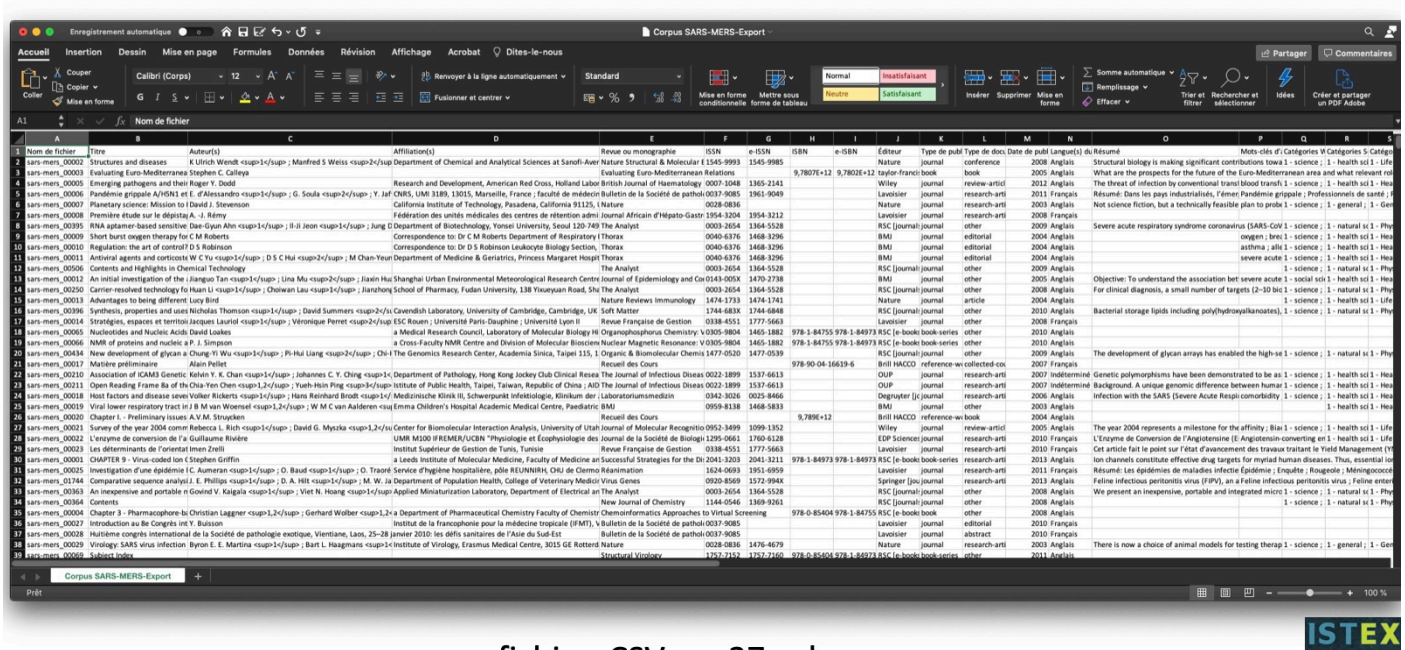
Aix*Marseille université
Socialement engagée

Exemple 1

Classification (partitionnement) de textes

Exemple : classification de textes

Nom de fichier;Titre;Auteur(s);Affiliation(s);Revue ou monographie;ISSN;e-ISSN;ISBN;e-ISBN;Éditeur;Type de publication;Type de document;Date d
r;Catégories WoS;Catégories Science-Metrix;Catégories Scopus;Catégories INIST;Score qualité;Version PDF;XML structuré;Identifiant ISTEX;ARK;DO
s_00002;Structures and diseases;"K Ulrich Wendt ¹ ; Manfred S Weiss ² ; Patrick Cramer ³ ; Dirk W Heinz <sup>
Sanofi-Aventis, Frankfurt, D-65926, Germany ; European Molecular Biology Laboratory, c/o DESY, Hamburg, D-22603, Germany ; Gene Centre, Ludwig
of Structural Biology, Helmholtz Centre for Infection Research, Braunschweig, D-38124, Germany";Nature Structural & Molecular Biology;1545-99
tural biology is making significant contributions toward an understanding of molecular constituents and mechanisms underlying human diseases a
rnau Conference on Structural Biology of Disease Mechanisms held in September 2007 in Murnau, Germany.";;"1 - science ; 2 - cell biology ; 2 -
lth sciences ; 2 - biomedical research ; 3 - developmental biology";"1 - Life Sciences ; 2 - Biochemistry, Genetics and Molecular Biology ; 3
Genetics and Molecular Biology ; 3 - Structural Biology";1 - sciences humaines et sociales;5.26;1.4;Absent;C20103CC68E46DEBF1A30871D342FC7F50B
7



fichier .CSV sur 27 colonnes
(méta-données ISTEX)

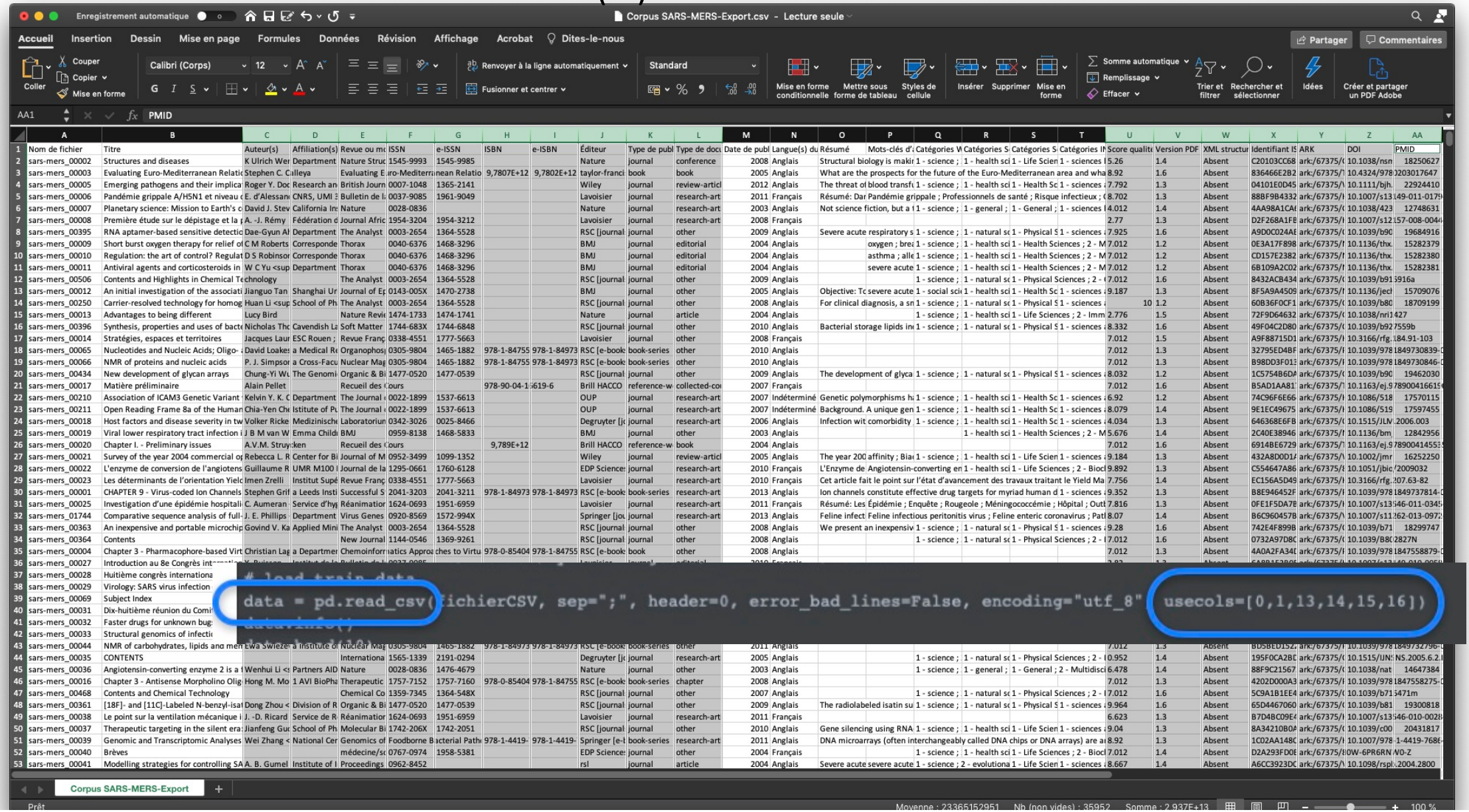
```
import pandas as pd

fichierCSV = "//Users/Patrice/PycharmProjects/ANF2021/ANF/test2.csv"
# load train data
data = pd.read_csv(fichierCSV, sep=";", header=0, error_bad_lines=False, encoding="utf_8")
data.head(10)
```

	Nom de fichier	Titre	Auteur(s)	Affiliation(s)	Revue ou monographie	ISSN	e-ISSN	ISBN	e-ISBN	Éditeur	...	Catégories Science-Metrix	Catégories Scopus	Catégories INIST	S qu
0	sars-mers_00002	Structures and diseases	K Ulrich Wendt ¹ ; Manfred S Weiss ...	Department of Chemical and Analytical Sciences...	Nature Structural & Molecular Biology	1545-9993	1545-9985	NaN	NaN	Nature	...	1 - health sciences ; 2 - biomedical research ...	1 - Life Sciences ; 2 - Biochemistry, Genetics...	1 - sciences humaines et sociales	5.
1	sars-mers_00003	Evaluating Euro-Mediterranean Relations	Stephen C. Calleya	NaN	Evaluating Euro-Mediterranean Relations	NaN	NaN	9.780715e+12	9.780203e+12	taylor-francis	...	NaN	NaN	NaN	8.
2	sars-mers_00005	Emerging pathogens and their implications for ...	Roger Y. Dodd	Research and Development, American Red Cross, ...	British Journal of Haematology	0007-1048	1365-2141	NaN	NaN	Wiley	...	1 - health sciences ; 2 - clinical medicine ; ...	1 - Health Sciences ; 2 - Medicine ; 3 - Hemat...	1 - sciences appliquees, technologies et medec...	7.
3	sars-mers_00006	Pandémie grippale A/H5N1 et niveau de préparat...	E. d'Alessandro ¹ ; G. Soula <sup>2...	CNRS, UMI 3189, 13015, Marseille, France ; fac...	Bulletin de la Société de pathologie exotique	0037-9085	1961-9049	NaN	NaN	Lavoisier	...	NaN	NaN	NaN	8.
4	sars-mers_00007	Planetary science: Mission to Earth's core — a...	David J. Stevenson	California Institute of Technology, Pasadena, ...	Nature	0028-0836	NaN	NaN	NaN	Nature	...	1 - general ; 2 - general science & technology...	1 - General ; 2 - Multidisciplinary ; 3 - Mult...	1 - sciences humaines et sociales	4.
5	sars-mers_00008	Première étude sur le dépistage et la prise en...	A. -J. Rémy	Fédération des unités médicales des centres de...	Journal Africain d'Hépato-Gastroentérologie	1954-3204	1954-3212	NaN	NaN	Lavoisier	...	NaN	NaN	NaN	2.
6	sars-mers_00395	RNA aptamer-based sensitive detection of	Dae-Gyun Ahn ¹	Department of Biotechnology, Yonsei	The Analyst	0003-2654	1364-5528	NaN	NaN	RSC [journals]	...	1 - natural sciences ; 2 - chemistry ;	1 - Physical Sciences ; 2 - Chemistry ; 3 -	1 - sciences appliquees, technologies	7.

data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 27 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Nom de fichier              9 non-null      object
 1   Titre                       9 non-null      object
 2   Auteur(s)                   9 non-null      object
 3   Affiliation(s)              8 non-null      object
 4   Revue ou monographie        9 non-null      object
 5   ISSN                        8 non-null      object
 6   e-ISSN                      7 non-null      object
 7   ISBN                        1 non-null      float64
 8   e-ISBN                      1 non-null      float64
 9   Éditeur                     9 non-null      object
 10  Type de publication         9 non-null      object
 11  Type de document            9 non-null      object
 12  Date de publication         9 non-null      int64
 13  Langue(s) du document       9 non-null      object
 14  Résumé                      6 non-null      object
 15  Mots-clés d'auteur          4 non-null      object
 16  Catégories WoS              6 non-null      object
 17  Catégories Science-Metrix   6 non-null      object
 18  Catégories Scopus           6 non-null      object
 19  Catégories INIST            4 non-null      object
 20  Score qualité               9 non-null      float64
 21  Version PDF                 9 non-null      float64
 22  XML structuré               9 non-null      object
 23  Identifiant ISTEX           9 non-null      object
 24  ARK                         9 non-null      object
 25  DOI                         9 non-null      object
 26  PMID                        6 non-null      float64
dtypes: float64(5), int64(1), object(21)
```

Identifier les colonnes (in)utiles

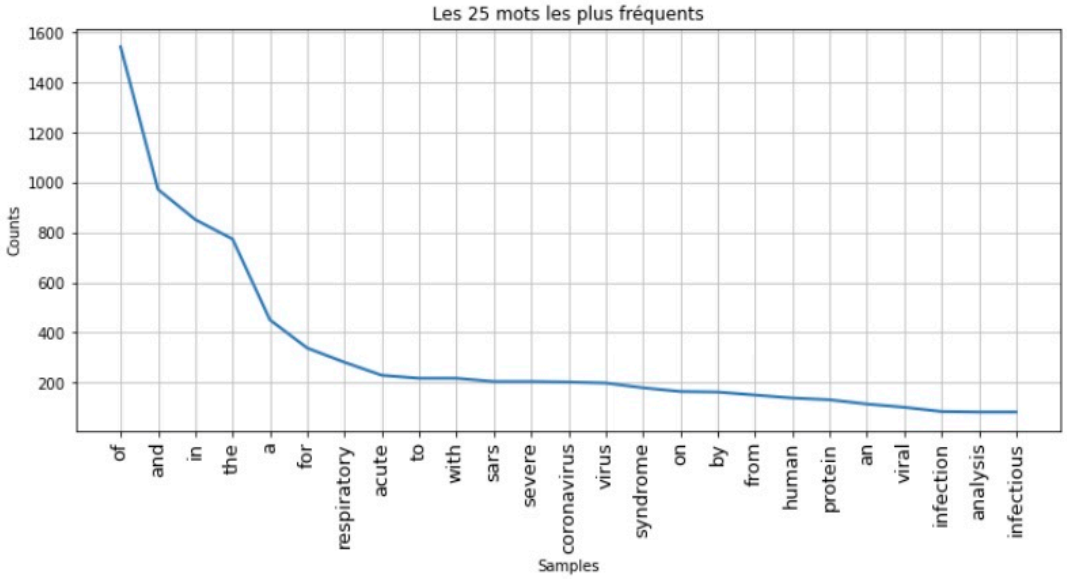


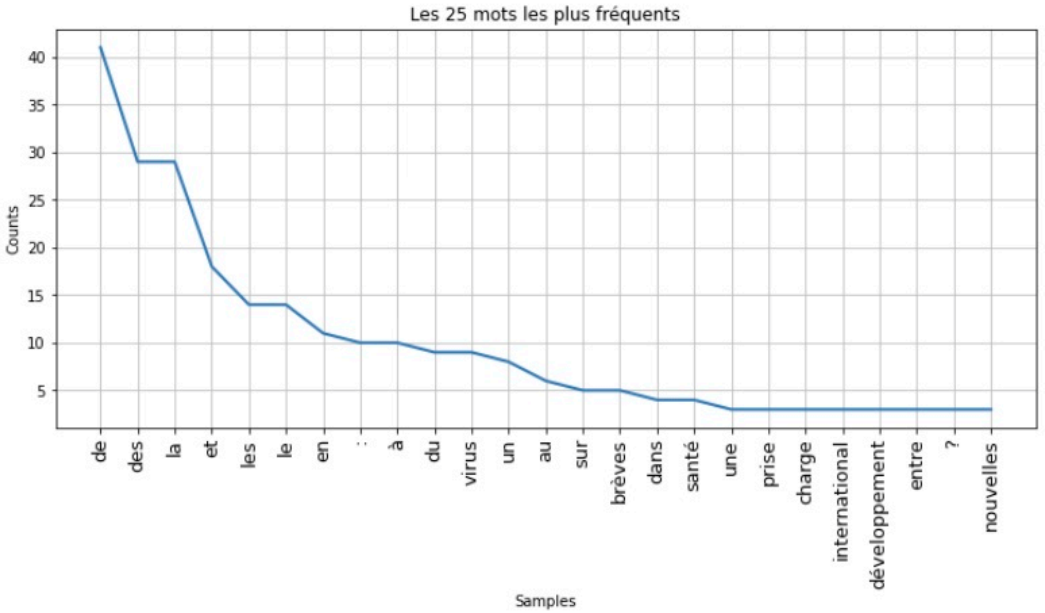
```
print('Nombre total de documents :  ',len(data), data.shape)  data: {DataFrame:
#Nombre de document par langue
print("Nombre de documents en français : ", len(data[data.Langue=='Français']))
for langue in data['Langue'].unique():  data: {DataFrame: (2532, 6)}
    print("Documents en ", langue, " : ", len(data[data.Langue==langue]))  langu
data.groupby('Langue', dropna=False).describe()  data: {DataFrame: (2532, 6)}
```

```
for langue in data['Langue'].unique():  data: {DataFrame: (2532, 6)}
    mots_des_titres = []
    for titre in list(data['Titre'][data.Langue==langue]):  data: {DataFrame: (2532
        mots = titre.split()  titre: Trends in der Impfstoffentwicklung. DNA- und z
        for mot in mots:  mots: ['FÜr', 'eine', 'Reihe', 'von', 'Infektionskrankhei
            mots_des_titres.append(mot.lower())  mots_des_titres: ['wirksamkeit', '
    print("pour langue ", langue, " : ", Counter(mots_des_titres).most_common(25))
```

```
Nombre total de documents :   2532 (2532, 6)
Nombre de documents en français :  67
Documents en  Anglais  :  2197
Documents en  Français  :  67
Documents en  Indéterminé  :  230
Documents en  Allemand  :  38
Les mots les plus fréquents par langue dans les titres :
pour langue  Anglais  :  [('of', 1543), ('and', 973), ('in', 852), ('the', 774), ('a', 451), ('for', 338), ('respir
atory', 281), ('acute', 229), ('to', 217), ('with', 217), ('sars', 204), ('severe', 204), ('coronavirus', 202), ('v
irus', 198), ('syndrome', 179), ('on', 164), ('by', 162), ('from', 150), ('human', 138), ('protein', 131), ('an', 1
14), ('viral', 101), ('infection', 84), ('analysis', 82), ('infectious', 82)]
pour langue  Français  :  [('de', 41), ('des', 29), ('la', 29), ('et', 18), ('les', 14), ('le', 14), ('en', 11), ('
:', 10), ('à', 10), ('du', 9), ('virus', 9), ('un', 8), ('au', 6), ('sur', 5), ('brèves', 5), ('dans', 4), ('santé'
, 4), ('une', 3), ('prise', 3), ('charge', 3), ('international', 3), ('développement', 3), ('entre', 3), ('?', 3),
('nouvelles', 3)]
pour langue  Indéterminé  :  [('of', 170), ('and', 112), ('in', 112), ('the', 90), ('respiratory', 85), ('acute', 7
4), ('severe', 65), ('a', 57), ('with', 52), ('syndrome', 49), ('coronavirus', 49), ('for', 39), ('human', 37), ('b
y', 30), ('infection', 25), ('to', 23), ('viral', 20), ('virus', 17), ('patients', 17), ('influenza', 16), ('from',
14), ('clinical', 13), ('on', 12), ('disease', 12), ('is', 12)]
```

```
# plot word frequency distribution of first few words
plt.figure(figsize=(12,5))
plt.title('Les 25 mots les plus fréquents')
plt.xticks(fontsize=13, rotation=90)
fd = nltk.FreqDist(mots_des_titres)
fd.plot(25,cumulative=False)
```





Suppression des mots outils ? (*stopwords*)

```
(base) [ Patrice Mac-Pro-de-Patrice-4 ~/nltk_data ] ls corpora/stopwords
README  azerbaijani  dutch    finnish  german  hungarian   italian  nepali      portuguese  russian  spanish  tajik
arabic  danish       english  french   greek   indonesian  kazakh   norwegian   romanian    slovene  swedish  turkish
(base) [ Patrice Mac-Pro-de-Patrice-4 ~/nltk_data ]
```

```
au
aux
avec
ce
ces
dans
de
des
du
elle
en
et
eux
il
ils
je
la
le
les
leur
lui
ma
mais
me
même
mes
moi
mon
ne
nos
notre
nous
```

```
i
me
my
myself
we
our
ours
ourselv
you
you're
you've
you'll
you'd
your
yours
yoursel
yoursel
he
him
his
himself
she
she's
her
hers
herself
it
it's
its
itself
they
them
```

```
de
la
que
el
en
y
a
los
del
se
las
por
un
para
con
no
una
su
al
lo
como
más
pero
sus
le
ya
o
este
sí
porque
esta
entre
```

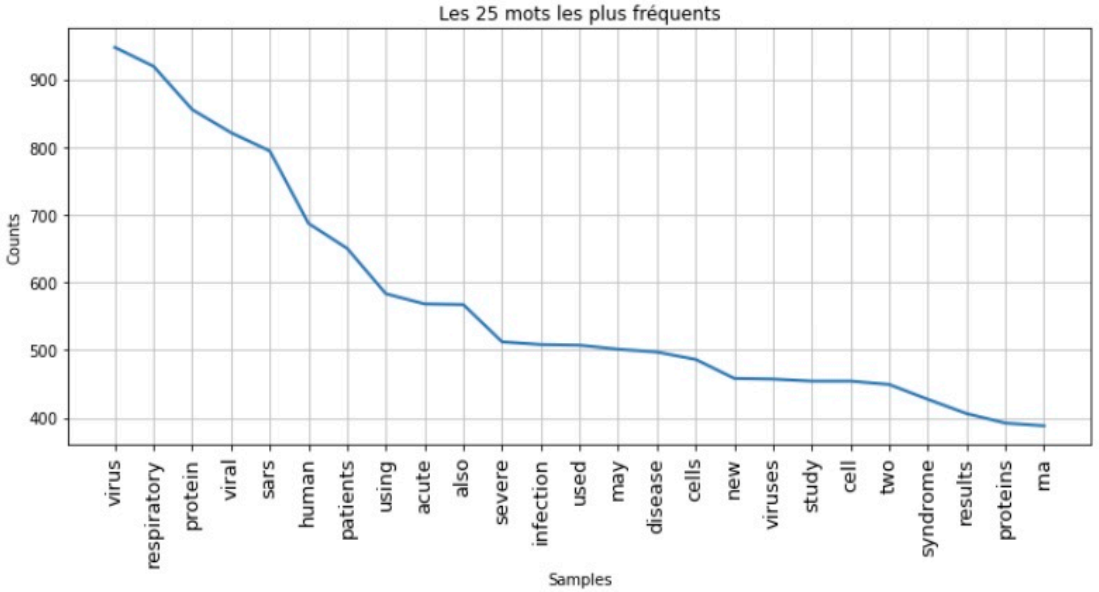
```
aber
alle
allem
allen
aller
alles
als
also
am
an
ander
andere
anderem
anderen
anderer
anderes
anderm
andern
anderr
anders
auch
auf
aus
bei
bin
bis
bist
da
damit
dann
der
den
```

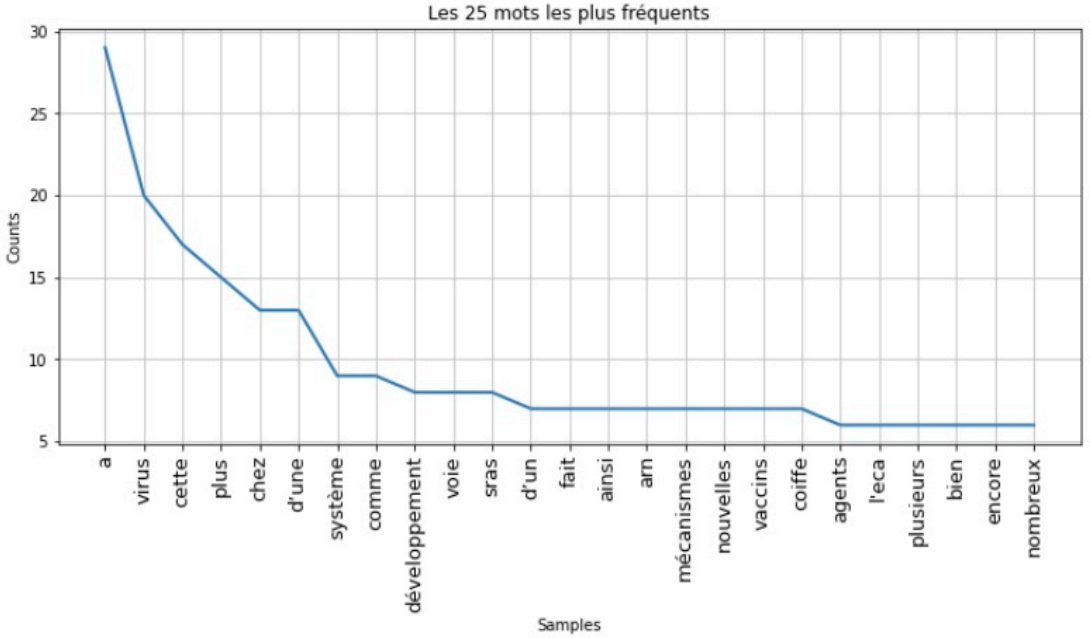
```
إذ
إذا
إذما
إذن
أف
أقل
أكثر
ألا
إلا
التي
الذي
الذين
اللاتي
اللائي
اللتان
اللتيا
اللتين
اللذان
اللذين
اللواتي
إلى
إليك
إليكم
إليكما
إليكن
أم
أما
أما
إما
أن
إن
```

```
и
в
во
не
что
он
на
я
с
со
как
а
то
все
она
так
его
но
да
ты
к
у
же
вы
за
бы
по
только
ее
мне
было
вот
```

```
 248 arabic
 164 azerbaijani
  94 danish
 101 dutch
 179 english
 235 finnish
 157 french
 232 german
 265 greek
 199 hungarian
 757 indonesian
 279 italian
 380 kazakh
 254 nepali
 176 norwegian
 204 portuguese
 355 romanian
 151 russian
1784 slovene
 313 spanish
 114 swedish
 162 tajik
  53 turkish
```

```
import nltk
from nltk.corpus import stopwords


for langue in data['Langue'].unique():
    mots_des_resumes = []
    stop_words = []
    if langue.lower() == "anglais":
        stop_words = stopwords.words('english')
    elif langue.lower() == "français":
        stop_words = stopwords.words('french')
    else:
        stop_words = stopwords.words('german')
    for texte in data['Resume'][data.Langue==langue]:
        if isinstance(texte,str):
            mots = texte.split()
            for mot in mots:
                if mot.lower() not in stop_words:
                    mots_des_resumes.append(mot.lower())
    print("pour langue ", langue, " : ", Counter(mots_des_resumes).most_common(25))
```


Les 25 mots les plus fréquents
Counts
900
800
700
600
500
400
virus
respiratory
protein
viral
sars
human
patients
using
acute
also
severe
infection
used
may
disease
cells
new
viruses
study
cell
two
syndrome
results
proteins
ma
Samples


Les 25 mots les plus fréquents
Counts
30
25
20
15
10
5
a
virus
cette
plus
chez
d'une
système
comme
développement
voie
sras
d'un
fait
ainsi
arn
mécanismes
nouvelles
vaccins
coiffe
agents
l'eca
plusieurs
bien
encore
nombreux
Samples

Les mots les plus importants ? (TF-IDF)

```
from sklearn.feature_extraction.text import TfidfVectorizer

def tfidf(textes):
    tfidfVec = TfidfVectorizer(stop_words='english', use_idf=True, min_df=2,
token_pattern='[a-z]+')
    tfidfMatrice = tfidfVec.fit_transform(textes)
    print("Les 10 premiers mots : ", tfidfVec.get_feature_names()[:10])
    return tfidfMatrice

tfidfMatrice = tfidf(resumesClasses['Resume'])

print("Taille matrice : ", tfidfMatrice.shape)
print("Représentation 1er résumé :", tfidfMatrice[0,:])
```

1276 documents, 7455 mots différents retenus, 1 vecteur par doc.

```
Les 10 premiers mots :  ['aa', 'ab', 'aberrant', 'abi', 'ability', 'able', 'abnormal', 'abnormalities', 'abo', 'abolish']
Taille matrice :  (1276, 7455)
Représentation 1er résumé :   (0, 2868) 0.2704313250945887
  (0, 6132)     0.24806386876009914
  (0, 3041)     0.23801900472661866
  (0, 1947)     0.09655372213525518
  (0, 1341)     0.2819297787345764
  (0, 3595)     0.1971843103507599
  (0, 1943)     0.17111132343868535
  (0, 5843)     0.17369088353073145
  (0, 533)      0.24272671127061593
  (0, 1949)     0.12103925328022304
  (0, 3202)     0.08852133361972166
  (0, 7069)     0.18858645715511513
  (0, 4125)     0.29376898845165716
  (0, 1403)     0.25422516491060365
  (0, 4331)     0.12637641076970627
  (0, 7072)     0.1328561891548727
  (0, 1457)     0.23801900472661866
  (0, 6230)     0.1291923149739438
  (0, 4024)     0.20415309475214133
  (0, 737)      0.35101596641375654
  (0, 6505)     0.2859850323557227
```

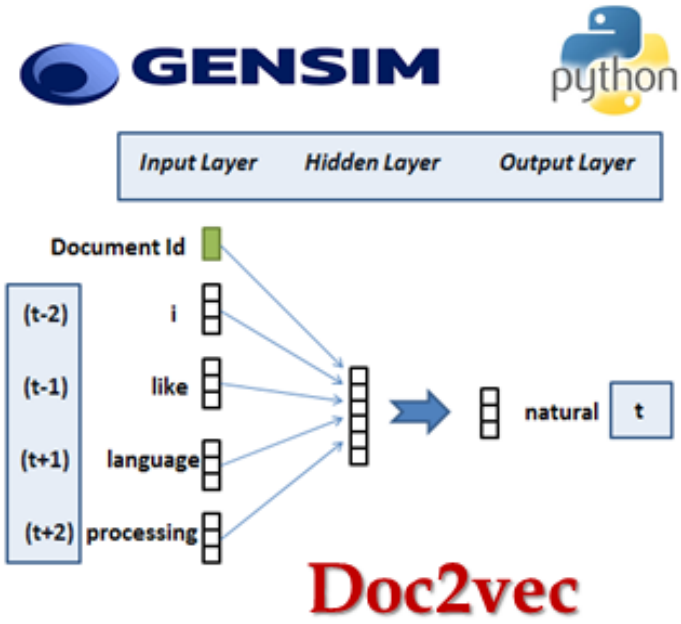
scores tf.idf

n° mot

Représentation en espaces réduits

Espace initial en très grande dimension (la taille du vocabulaire) :

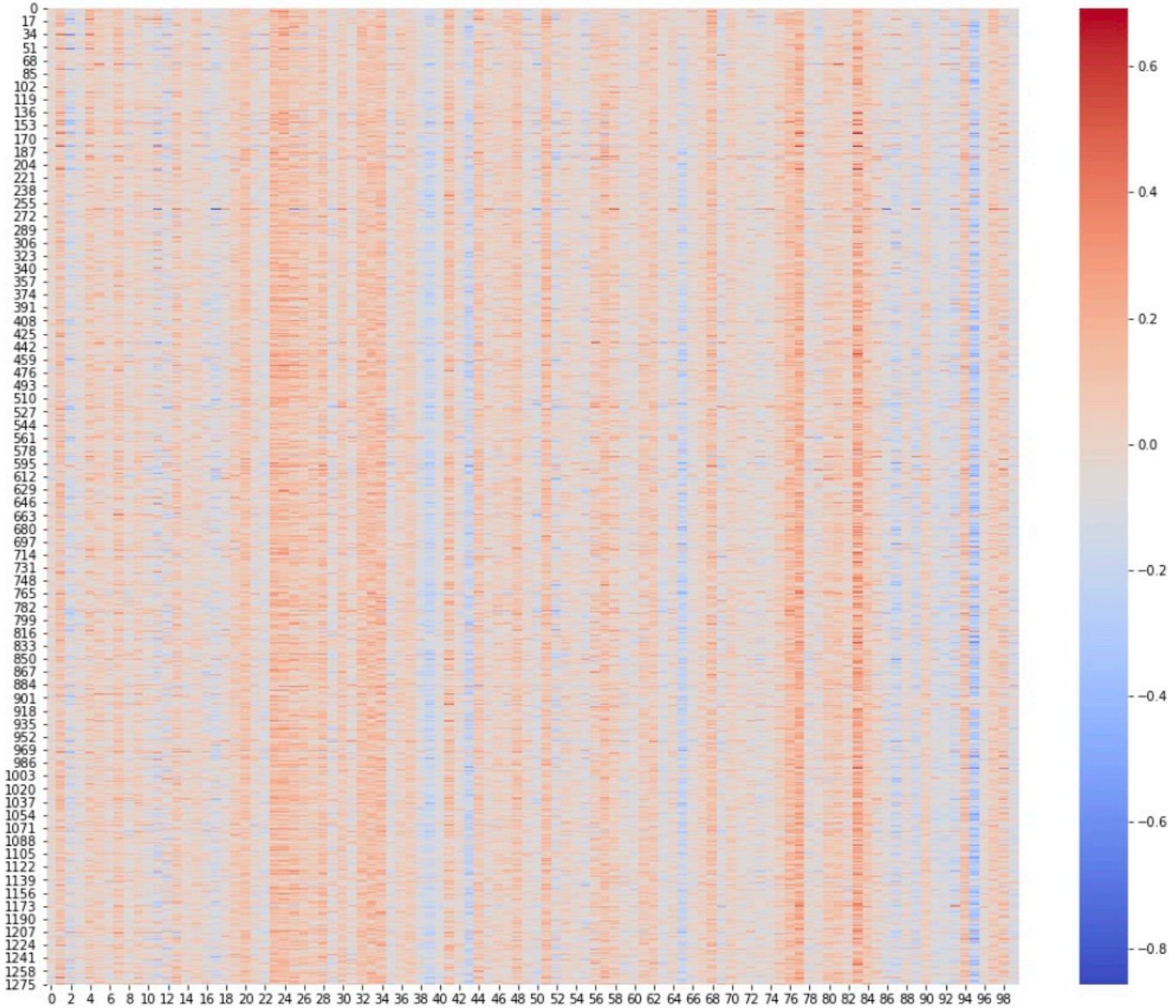
— réunir les mots similaires = projeter les documents sur un espace réduit




```
from gensim.models import Doc2Vec

def doc2vect():
    document_tagged  = []
    tagged_count = 0
    for _ in resumesClasses['Resume'].values:
      document_tagged.append(gensim.models.doc2vec.TaggedDocument(_,
[tagged_count]))
        tagged_count += 1
    d2v = Doc2Vec(document_tagged)
    return d2v.docvecs.vectors_docs

%time doc2vec = doc2vect()

import seaborn as sn
plt.figure(figsize=(17,14))
sn.heatmap(doc2vec, cmap = "coolwarm")
```

```
CPU times: user 6.05 s, sys: 252 ms, total: 6.3 s
Wall time: 3.01 s
```

6 `<AxesSubplot:>`

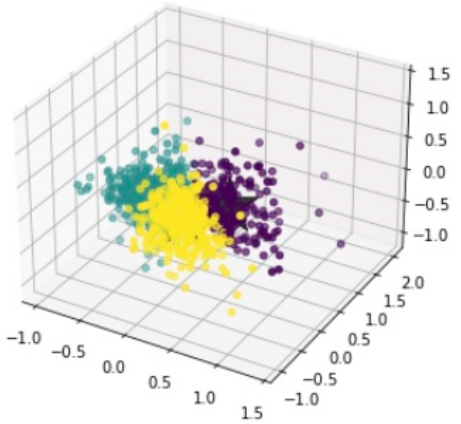


Classification non supervisée (k-moyennes)

```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.cm
import numpy as np

kmean_model = KMeans(n_clusters=3, n_jobs=-1)
%time km = kmean_model.fit_predict(doc2vec)
print ("intertie intra-classes : ", kmean_model.inertia_)
```

```
9  print(kmean_model.labels_)

   [0 1 2 ... 1 1 0]
```

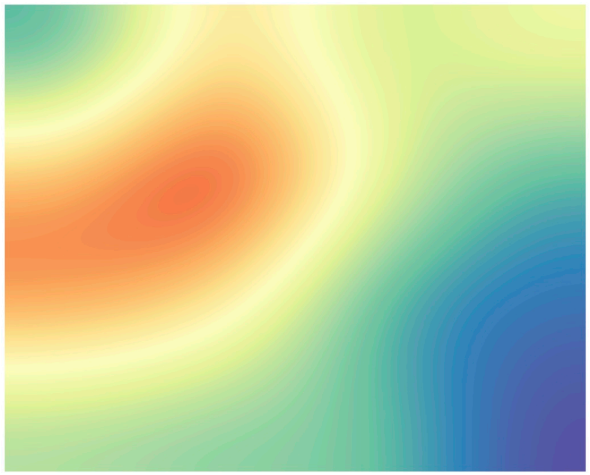
```
pca = PCA(n_components=3).fit(doc2vec)
data = pca.transform(doc2vec)
centroids = pca.transform(kmean_model.cluster_centers_)

#color = matplotlib.cm.rainbow(np.linspace(0, 1,
len(kmean_model.labels_)))
color = kmean_model.labels_
plt.figure(figsize=(50,20))
axis = Axes3D(plt.figure())
axis.scatter(data[:, 0], data[:, 1],data[:, 2],c =
color)
axis.scatter(centroids[:, 0], centroids[:, 1],
centroids[:,2], marker='*', s=1500, c='#000000')
plt.show()
```



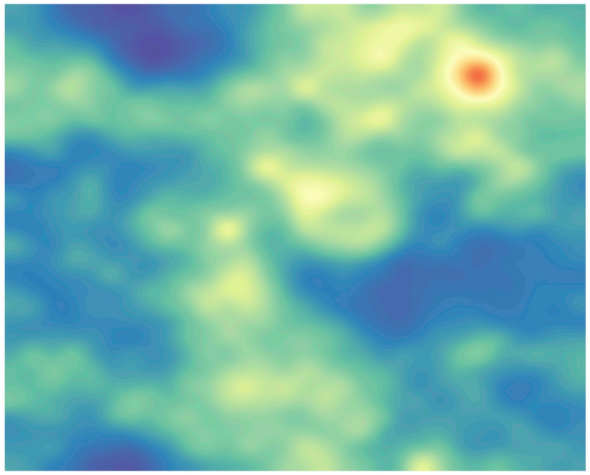
```
lt.figure(figsize=(17,14))
plt.scatter(data[:, 0], data[:, 1],c = color)
plt.scatter(centroids[:, 0], centroids[:, 1],
marker='*', s=200, c='#000000')
plt.show()
```

Cartographie documentaire (cartes auto-organisées)

```
import somoclu

som = somoclu.Somoclu(4, 4, maptype="toroid")
som.train(doc2vec)
som.view_umatrix(bestmatches=False,
figsize=(17,14))
```



```
som = somoclu.Somoclu(30, 30, maptype="toroid")
som.train(doc2vec)
som.view_umatrix(bestmatches=False,
figsize=(17,14))
```



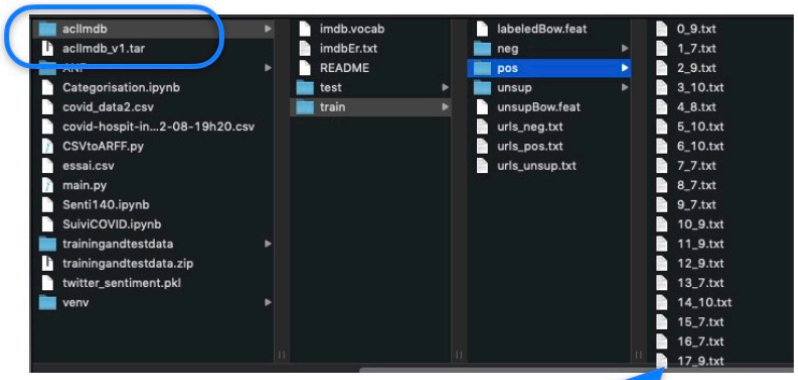
Aix*Marseille université
Socialement engagée

Exemple 2

Vers une analyse de sentiments automatisée

Large Movie Review Dataset

http://ai.stanford.edu/~amaas/data/sentiment/



Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011, June). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* (pp. 142-150).

4.3.2 IMDB Review Dataset

We constructed a collection of 50,000 reviews from IMDB, allowing no more than 30 reviews per movie. The constructed dataset contains an even number of positive and negative reviews, so randomly guessing yields 50% accuracy. Following previous work on polarity classification, we consider only highly polarized reviews. A negative review has a score ≤ 4 out of 10, and a positive review has a score ≥ 7 out of 10. Neutral reviews are not included in the dataset. In the interest of providing a benchmark for future work in this area, we release this dataset to the public.[2]

Corpus d'entraînement (train) : 12 500 positives, 12 500 négatives

```
This is a complex film that explores the effects of Fordist and Taylorist modes of industrial capitalist production on human relations. There are const
 references to assembly line production, where workers are treated as cogs in a machine, overseen by managers wielding clipboards, controlling how much
ir the workers leave exposed, and firing workers (Stanley) who meet all criteria (as his supervisor says, are always on time, are hard workers, do good
rk) but who may in some unspecified future make a mistake. <br /><br />This system destroys families - Stanley has to send his father to a nursing home
here he quickly dies) after Stanley loses his job. Iris' daughter is a single teen mother who drops out of high school to take a job in the plant. Refe
ces are made to the fact that now, with declining wages, both partners need to work, the implication being that there's nobody left at home to care for
e kids. Iris' husband is dead from an illness, and with the multiple references in the film about the costs of medical care, the viewer must wonder if
might have lived with better and more costly care. Iris' brother in law gets abusive after yet another unsuccessful day at the unemployment office when
s wife yells at him for buying a beer with her savings instead of leaving it for her face lift and/or teeth job (even the working class with no stake i
onventional bourgeois notions of perfection and beauty buy into them). The one reference to race in the film is through a black factory line worker who
husband is in jail (presumably, he's also black, and black men suffer disproportionally high incarceration rates). She remarks that he, like her, "is d
g time" - her family is composed of a prisoner and a wage slave.<br /><br />Stanley, however, still believes in human relations and is therefore for mo
of the film outside of the system of Fordist capitalism. He cares for his father in spite of the fact that it was his father's traveling salesman job t
 resulted in his illiteracy - he has not yet reduced human relations to a purely instrumental contract, as Iris' brother in law does (suggesting that h
married the wrong sister"). He does not, as Iris says, conform to the work-eat-sleep routine of everyone else; rather, he uses technology and the techn
es of industrial production in an artisanal and creative way, in a sort of Bauhaus ideal. This was the dream of early modernists and 1920's socialists
```

Pré-traitements du corpus

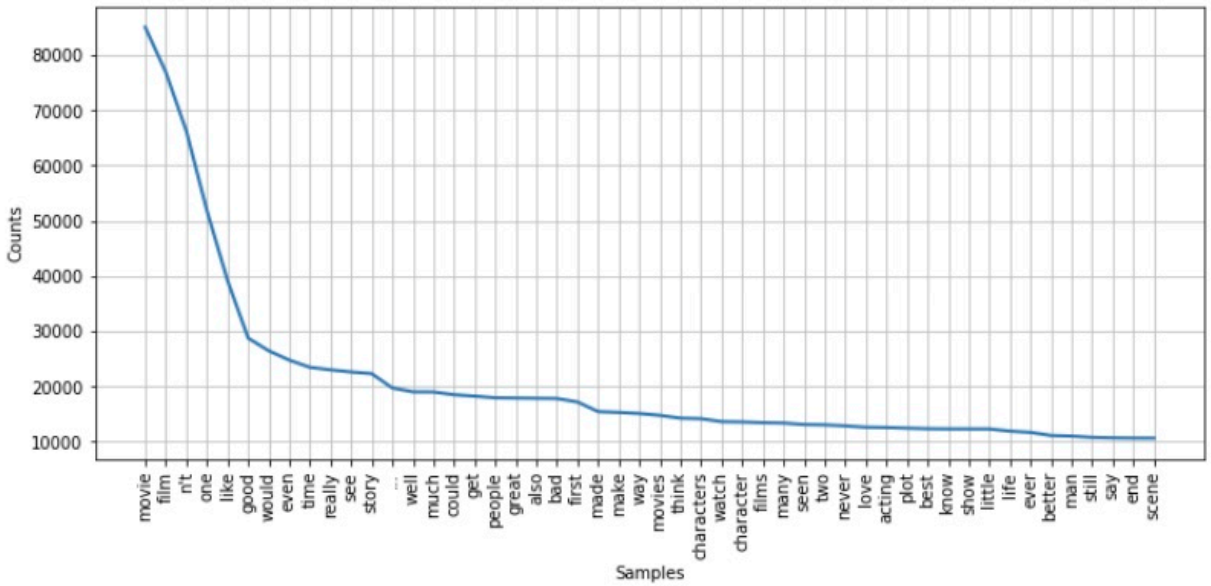
La première étape consiste à intégrer l'ensemble des critiques annotées (polarité négative ou positive) en un seul fichier au format CSV qui pourra être stocké en mémoire par un DataFrame (extension Pandas de Python).

taille du fichier movie_dataset.csv : 65,9 Mo (50 000 lignes, 14 millions de *tokens*, 194 758 mots différents)

```
# Conversion du corpus d'origine en un fichier .csv

import pandas as pd
import os

repertoire_depart = '/Users/Patrice/PycharmProjects/ANF2021/aclImdb'

labels = {'pos':1, 'neg' : 0}
df = pd.DataFrame()
for f in ('test', 'train'):
    for l in ('pos', 'neg'):
        path = os.path.join(repertoire_depart, f, l)
        for fichier in os.listdir(path):
            with open(os.path.join(path, fichier), 'r', encoding='utf-8') as infile:
                txt = infile.read()
            df = df.append([[txt, labels[l]]], ignore_index=True)
df.columns=['review', 'polarity']

df.to_csv('movie_data.csv', index=False, encoding='utf-8')
df.head()
```

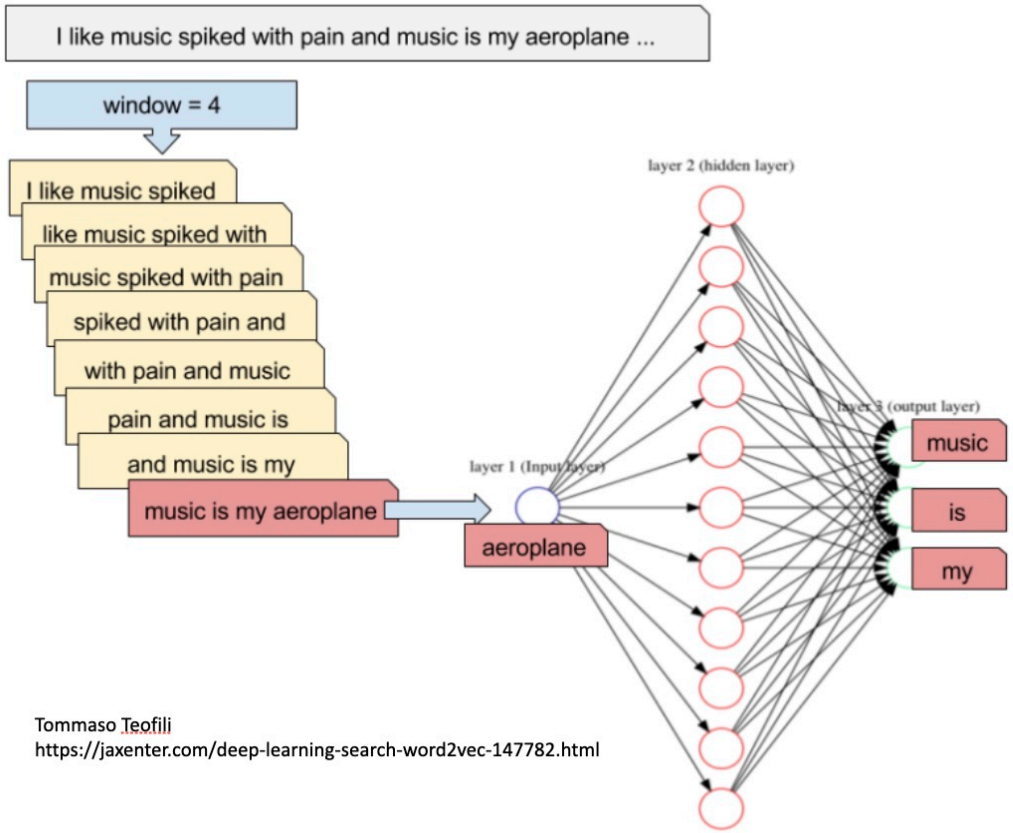
	review	polarity
0	Based on an actual story, John Boorman shows t...	1
1	This is a gem. As a Film Four production - the...	1
2	I really like this show. It has drama, romance...	1
3	This is the best 3-D experience Disney has at ...	1
4	Of the Korean movies I've seen, only three had...	1

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

#Concatenate strings in the Series/Index with given separator.
reviews = df.review.str.cat(sep=' ')
#function to split text into word
tokens = word_tokenize(reviews)
stop_words = set(stopwords.words('english'))
tokens = [w.lower() for w in tokens if not w.lower() in stop_words and len(w)>2]
vocabulary = set(tokens)
print("Taille du vocabulaire : ", len(vocabulary))
frequency_dist = nltk.FreqDist(tokens)
sorted(frequency_dist,key=frequency_dist.__getitem__, reverse=True)[0:50]
plt.figure(figsize=(12,5))
frequency_dist.plot(50)
```

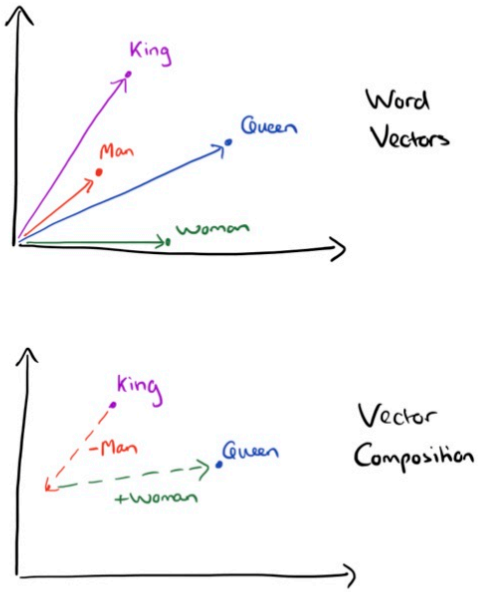


```python
from wordcloud import WordCloud
import matplotlib.pyplot as plt
reviewsPositives = df.review[df.polarity==1].str.cat(sep=' ')
tokensPos = word_tokenize(reviewsPositives)
tokensPos = [w.lower() for w in tokensPos if not w.lower() in stop_words and len(w)>2]
vocabPos = set(tokensPos)
frequency_dist_Pos = nltk.FreqDist(tokensPos)
sorted(frequency_dist_Pos,key=frequency_dist_Pos.__getitem__, reverse=True)[0:50]
wordcloud = WordCloud(width=1200, height=800,
                      max_words=500,
                      max_font_size=100,
                      relative_scaling=0.5,
                      colormap='Greens',
                      normalize_plurals=True).generate_from_frequencies(frequency_dist_Pos)
plt.figure(figsize=(17,14))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

Réduction de la dimension (projection)
Les plongements de mots (word embeddings)



Tommaso Teofili
https://jaxenter.com/deep-learning-search-word2vec-147782.html



Adrian Colyer
https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vector

Distributed Representations of Words and Phrases and their Compositionality – Mikolov et al. 2013

Efficient Estimation of Word Representations in Vector Space – Mikolov et al. 2013

Apprentissage de représentation Word2Vec

```
#L'espace de réprésentation est appris sur l'ensemble du corpus
for line in df['review']:
    tokens = word_tokenize(line)
    stop_words = set(stopwords.words('english'))
    tokens = [w.lower() for w in tokens if w.isalpha() and len(w)>1 and not w.lower() in stop_words]
    review_lines.append(tokens)
```

```
import gensim
model = gensim.models.Word2Vec(sentences=review_lines, size=200, window=5, workers=4, min_count=1)
motsComplet = list(model.wv.vocab)
```

pour chaque mot, un vecteur en 200 dimensions

```
movie -1.3216566 0.36635584 -0.28186616 -1.0511837 -1.0501945 -1.7482823 -0.42692444 0.16830114 -1.073119 -1.5651205 -
96654 -0.54516864 1.2929311 0.49605948 1.1482662 0.38361785 -0.30000296 0.78807664 -0.62371856 -1.5082116 -0.13787036 -
74925745 0.41954425 0.35796735 0.3195898 -0.20374134 -0.25748256 -0.90302813 -0.44684523 -0.46419883 0.43331063 0.3801
-0.23262957 -0.57022005 -0.6890808 0.29229978 -0.06665888 -0.045591816 -0.31439704 -0.44238204 -1.19862 0.12611166 0.9
1796 0.17370766 0.20563798 0.8580158 0.8143437 -0.026487244 -0.12953776 1.6001002 0.2723402 0.053601284 0.440381 0.058
9 -0.14719109 -0.3533582 -1.16035 1.0383319 0.3641711 -0.29797938 -0.041548226 0.35354558 -0.7025537 0.17
1.3149184 0.21495351 -0.7291604 0.18647747 -1.2000268 -0.51228637 0.36612657 -0.25129464 -0.746 -0.183673
6096396 0.34609687 0.40593633 0.7030198 0.023112642 -0.9067271 0.43155307 0.4280309 -0.049969178 -0.67905
.6962609 0.16017178 0.66016424 -0.5926901 -0.013376136 -0.22369754 -1.0953285 -0.56589377 -0.42723322 0.7
673262 0.8491248 -0.484025 -0.31997883 0.18664318 -0.5761222 0.33220634 -1.0463667 -0.009183551 0.5471651
037895 -1.0772457 -0.646116 1.1264194 -0.9413773 0.08854891 -0.122176886 -0.056594223 0.5072317 1.13529 -
88807 0.37230954 -0.61006385 -1.1492089 -1.5274029 -0.037806857 -0.19853547 0.2762417 -0.9356259 -0.37737
film -1.1342325 0.5424572 0.0140855415 -0.54681146 -1.0229077 -2.3149817 -0.3617721 0.08117554 -0.69557166 -1.0018283 -
25 0.1879876 1.2258501 0.53333026 0.71119124 -1.3764403 -0.69352823 0.67989963 0.049601056 -1.0814724 -0.17875 -0.2995
0.46457902 0.110982075 0.07333746 -1.321096 -0.19277126 0.023522813 -0.31523454 -0.23818257 0.4992599 0.20365019 0.210
204 0.43208042 0.03197141 0.19413853 -0.32528928 -0.14852582 0.12936993 0.068569176 -0.36599588 0.116247706 0.68026376
314871 -0.30278912 0.69517577 0.4294458 -0.3990693 -0.76446646 1.5112543 0.3708154 0.11746891 0.701029 -0.7823005 1.62
8 -0.26889926 1.0239882 -1.2052739 -0.047914516 0.9869529 -0.46331605 -0.07111113 0.079658456 0.37919065 -0.006453751 -
45773625 -0.58498067 0.45197055 -0.49910277 0.317274 -0.90511173 0.42767948 0.22158863 -0.068598926 0.58532935 -0.0108
21 2.0317261 0.7017311 0.12857646 1.0322477 0.30594614 0.5822884 -1.2792618 -0.27707702 0.5073626 0.5156112 -0.7731857
63963 -0.25596482 0.66147095 -0.007577596 -1.0135919 -0.37657994 0.21909198 -1.2694278 -0.758413 -0.9453872 -0.2356834
5475771 0.36981234 0.29823944 -0.37622204 0.22047852 0.2637362 -1.1235323 0.12577608 -0.56808615 -0.49570698 0.2905903
0.37622717 0.11014897 -1.2906862 0.10878839 0.9532716 -0.9014037 -0.41337353 0.57484233 -0.76305604 0.26593128 0.29173
```

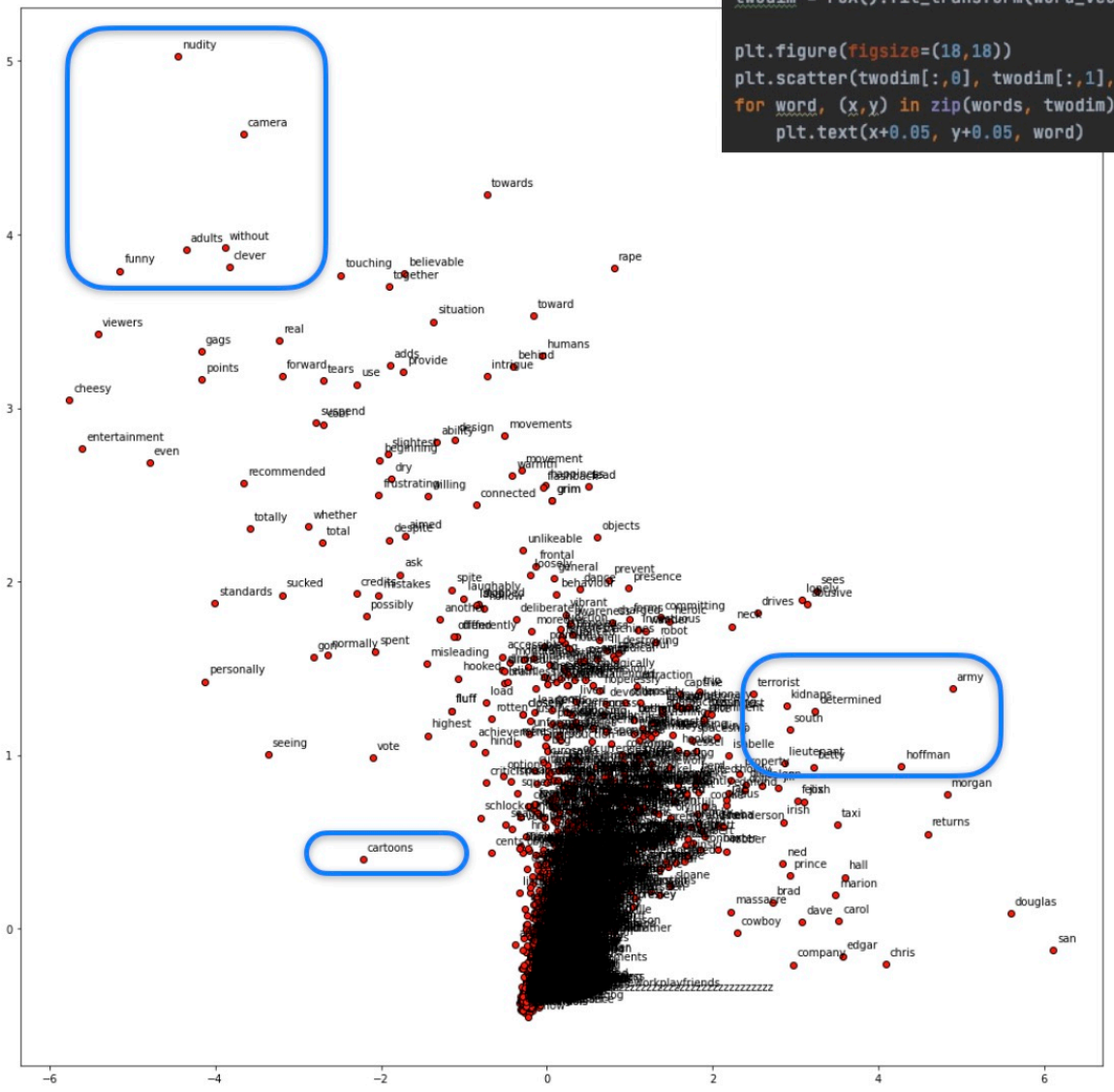
Aix*Marseille université
Socialement engagée

cnrs Aix*Marseille université

https://web.stanford.edu/class/cs224n/materials/Gensim%20word%20vector%20visualization.html

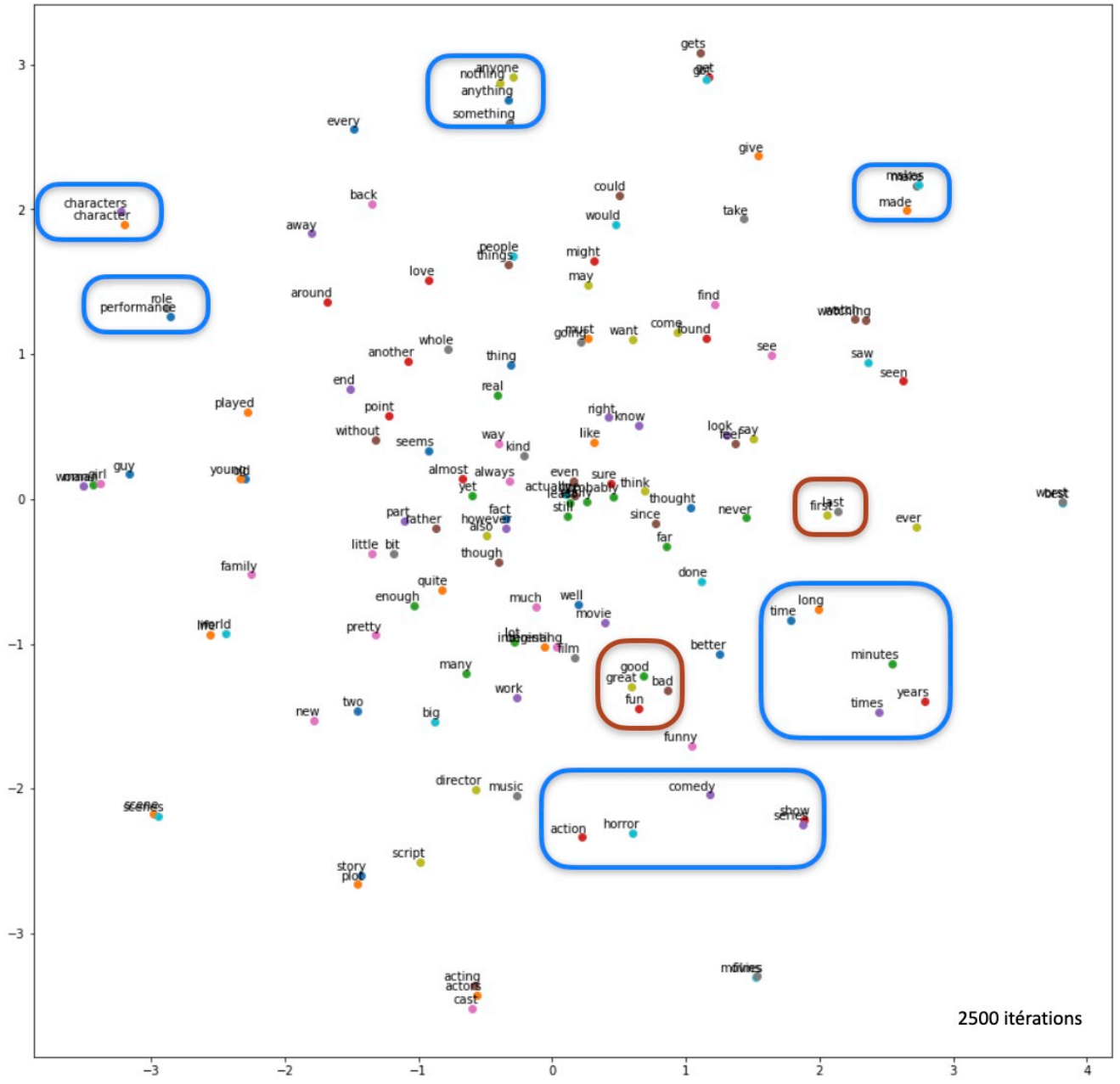
ACP
de 3000
mots pris
au hasard

```
twodim = PCA().fit_transform(word_vectors)[:,:2]

plt.figure(figsize=(18,18))
plt.scatter(twodim[:,0], twodim[:,1], edgecolors='k', c='r')
for word, (x,y) in zip(words, twodim):
    plt.text(x+0.05, y+0.05, word)
```



Aix*Marseille université
Socialement engagée

t-SNE des mots dont occurrences > 5000 en limitant Word2Vec aux mots ayant au moins 100 occurrences

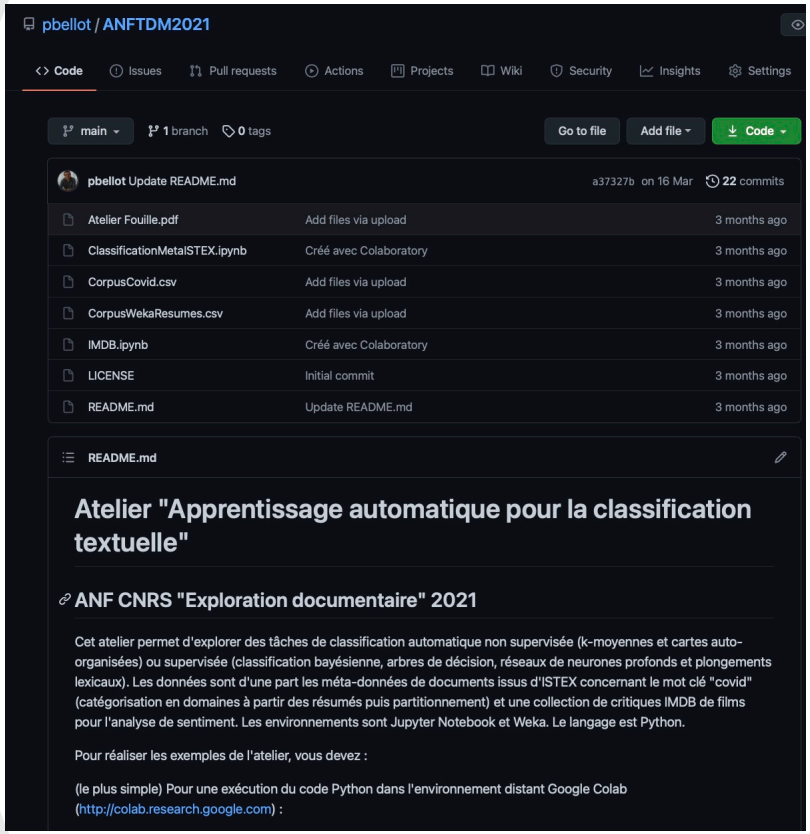


https://pub.towardsai.net/understanding-semantic-analysis-using-python-nlp-f48016422677

Démarche générale (pré-traitements)

```
new_df = pd.DataFrame({'document':df})

# removing everything except alphabets
new_df['clean_doc'] = new_df['document'].str.replace("[^a-zA-Z#]", " ")

# removing short words
new_df['clean_doc'] = new_df['clean_doc'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))

# make all text lowercase
new_df['clean_doc'] = new_df['clean_doc'].apply(lambda x: x.lower())
```

```
from nltk.corpus import stopwords
swords = stopwords.words('english')

# tokenization
tokenized_doc = new_df['clean_doc'].apply(lambda x: x.split())

# remove stop-words
tokenized_doc = tokenized_doc.apply(lambda x: [item for item in x if item not in swords])

# de-tokenization
detokenized_doc = []
for i in range(len(news_df)):
    t = ' '.join(tokenized_doc[i])
    detokenized_doc.append(t)

new_df['clean_doc'] = detokenized_doc
```



https://github.com/pbellot/ANFTDM2021

https://anf-tdm-2020.sciencesconf.org/329939

https://www.youtube.com/channel/UCFyKvV_dkY2ww2ejT4pcF1g

Des exemples de .csv

https://people.math.sc.edu/Burkardt/datasets/csv/csv.html
https://www.opendataphilly.org/dataset/parking-violations
https://www.kdnuggets.com/datasets/index.html
https://www.kaggle.com/datasets
https://perso.telecom-paristech.fr/eagan/class/igr204/datasets
https://www.dataquest.io/blog/free-datasets-for-projects/