Cloud-Words SAE

Présentation

Qu'est-ce qu'un Cloud Word ?

Un Cloud Word, aussi appelé nuage de mots clés, est une image représentant plusieurs mots prenant une dimension plus ou moins importante selon leur occurence dans un texte donné.

L'intérêt est de pouvoir visualiser les mots les plus utilisés et par conséquent les sujets et thèmes abordés dans le texte, dans une perspective d'analyse des données.

Dans le cas de notre étude

Le jeu de données sur lequel nous travaillons étant sur des articles scientifiques, on s'attend donc à obtenir des résultats orientés sur plusieurs domaines techniques, tels que des notions mathématiques, médicales.

In [ ]:
```
%pip install matplotlib wordcloud tensorflow tqdm
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-packages (1.9.2)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.14.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (4.66.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.45.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes==0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/
dist-packages (from tensorflow) (3.20.3)
```

```
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.34.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.59.3)
Requirement already satisfied: tensorboard<2.15,>=2.14 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.1)
Requirement already satisfied: tensorflow-estimator<2.15,>=2.14.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.0)
Requirement already satisfied: keras<2.15,>=2.14.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.42.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14->tensorflow)
(2.17.3)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14->tenso
rflow) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14->tensorflow) (3.5.1)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14->tensorflow) (2.3
1.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14
->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14->tensorflow) (3.0.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.1
5,>=2.14->tensorflow) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.1
5,>=2.14->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.15,>=2.14-
>tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<1.1,>=0.5->tens
orboard<2.15,>=2.14->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.1
5,>=2.14->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.15,>=2.14->te
nsorflow) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.15,>=2.
14->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.15,>=2.
14->tensorflow) (2023.11.17)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.15,>=2.14->t
ensorflow) (2.1.3)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=
1.6.3->tensorboard<2.15,>=2.14->tensorflow) (0.5.1)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthli
b<1.1,>=0.5->tensorboard<2.15,>=2.14->tensorflow) (3.2.2)
```

Mise en place du code

Importation des différentes bibliothèques Python

- json pour la manipulation des données au format JSON.

- matplotlib.pyplot pour la création de datavisualisations.
- wordcloud pour générer des nuages de mots à partir de texte.
- tensorflow pour récupérer notre jeu de données.
- time pour la gestion du temps, par exemple, pour mesurer la durée d'exécution d'une opération.
- pandas pour la manipulation et l'analyse des données.

In [ ]:
```
import json
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import tensorflow as tf
import time
import pandas as pd
```

Importation des datasets

Pour créer notre nuage de mots, nous importons trois ensembles de données différents : le jeu de données d'entraînement, celui de test et enfin celui de validation. Grâce aux nuages de mots que nous allons générer, nous pourrons rapidement visualiser la couverture globale des thèmes abordés dans l'ensemble des données, et ainsi les comparer entre eux

In [ ]:
```
from google.colab import drive
import json
import os
import sys
```

In [ ]:
```
drive.mount('/content/drive', force_remount=True)
sys.path.append('/content/drive/MyDrive/package')
import datasets_scientific_paper as load_ds
```

```
Mounted at /content/drive
Mounted at /content/drive
```

In [ ]:
```
pathToDataset = "drive/MyDrive/chunking-dataset"

train_data, train_labels = load_ds.load_dataset(pathToDataset, "train")
test_data, test_labels = load_ds.load_dataset(pathToDataset, "test")
val_data, val_labels = load_ds.load_dataset(pathToDataset, "val")
```

Création des Word Cloud

Dans un premier temps, pour créer nos nuages de mots, on crée deux fonctions au préalable.

Tout d'abord, la fonction createText, qui prendra un ensemble de données et un nombre correspondant à la taille de l'échantillon souhaité. Cette fonction parcourra tous les articles, les regroupera en un seul texte en les concaténant.

On créera également la fonction displayWordCloud, ayant pour seul effet d'afficher l'image générée par l'objet WordCloud de Python.

In [ ]:
```
def createText(dataset, sample_size):
    text = ""
    for article in dataset[:sample_size]:
        article_text = article.get('article_text', [])
        text += ' '.join(article_text) + ' '
    return text

def displayWordCloud(wordcloud):
    plt.figure(figsize=(15,8))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()
```

Ici, on appelle la fonction createText que l'on stocke dans une variable afin de pouvoir l'utiliser par la suite. On précise également une taille d'échantillon pour éviter de consommer trop de ressources.

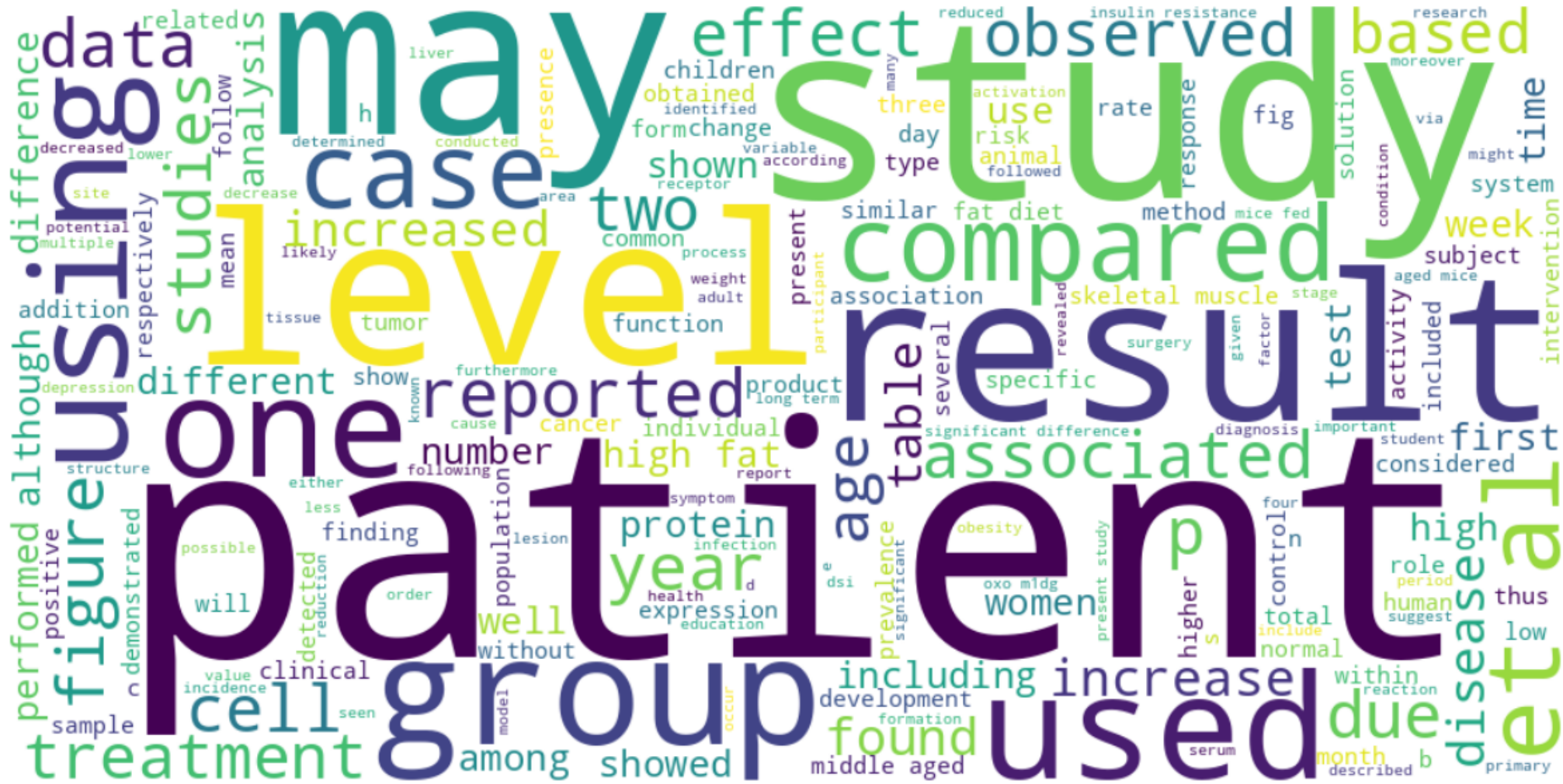
In [ ]:
```
sampleSize = 100
textTrain = createText(train_data,sampleSize)
textTest = createText(test_data,sampleSize)
textVal = createText(val_data,sampleSize)
```

on génère à cette étape nos nuages de mots pour les afficher par la suite.

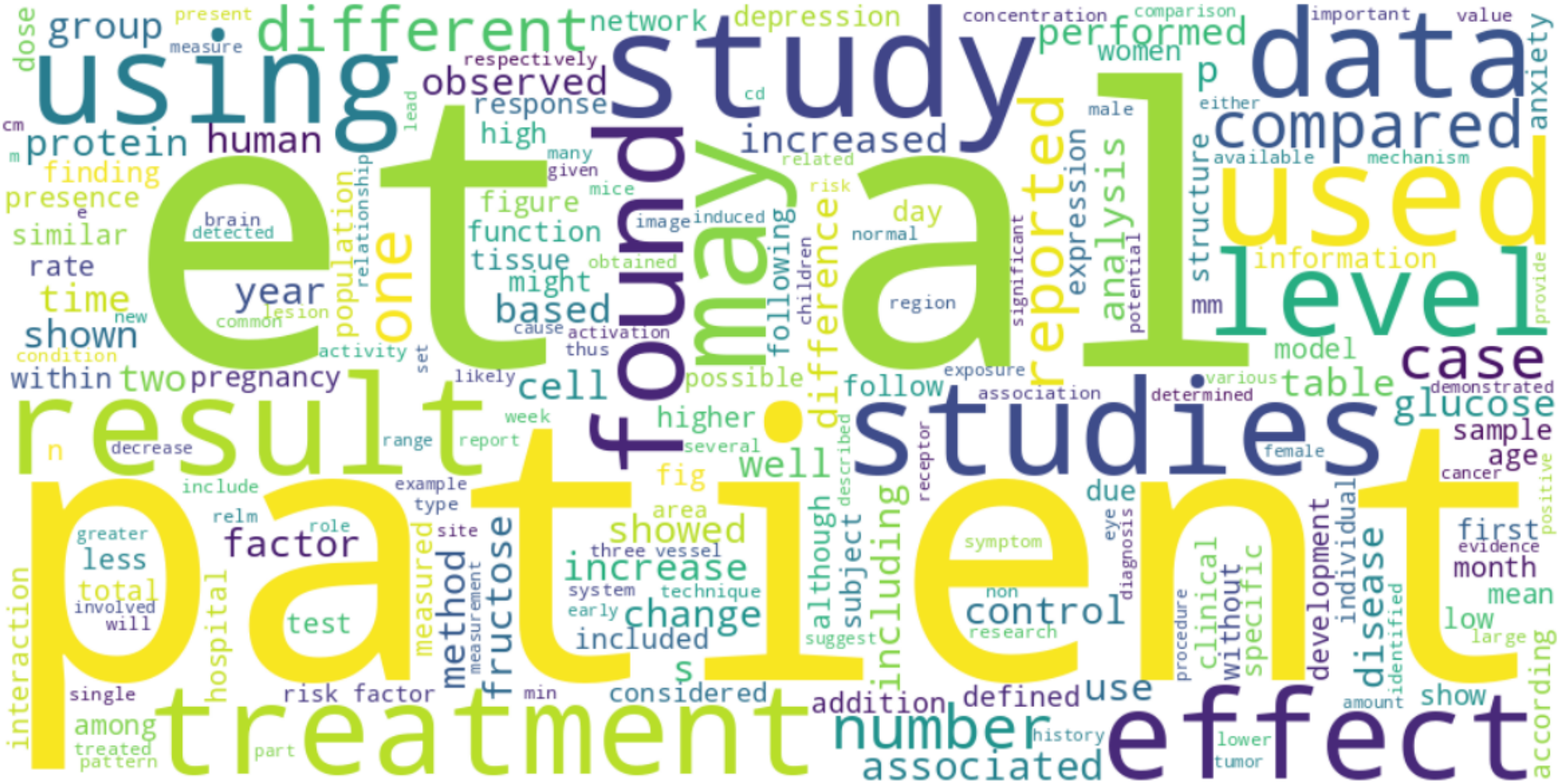
In [ ]:
```
worldCloudTrain = WordCloud(width=1000, height=500,background_color='white', colormap='viridis').generate(textTrain)
worldCloudTest = WordCloud(width=1000, height=500,background_color='white', colormap='viridis').generate(textTest)
worldCloudVal = WordCloud(width=1000, height=500,background_color='white', colormap='viridis').generate(textVal)
```

In [ ]:
```
print("Nuage de Mots : Train")
displayWordCloud(worldCloudTrain)
print("Nuage de Mots : Test")
displayWordCloud(worldCloudTest)
print("Nuage de Mots : Val")
displayWordCloud(worldCloudVal)
```

```
Nuage de Mots : Train
```



Nuage de Mots : Test



Nuage de Mots : Val

Analyse des données

Maintenant que l'on a créé et affiché nos différents nuages de mots, on peut procéder à une étape importante qui est l'analyse des données. Pour ce faire, on peut utiliser WordCloud.words_ pour récupérer dans un dictionnaire Python les mots et leurs fréquences.

Grâce à cette information, on crée par la suite deux fonctions en Python ayant pour but de réaliser de la datavisualisation.

La première, wordFrequency, permettant de générer un graphique avec les fréquences de chaque mot dans les différents textes. Plus le chiffre se rapproche de 1, plus le mot apparaîtra grand dans le WordCloud.

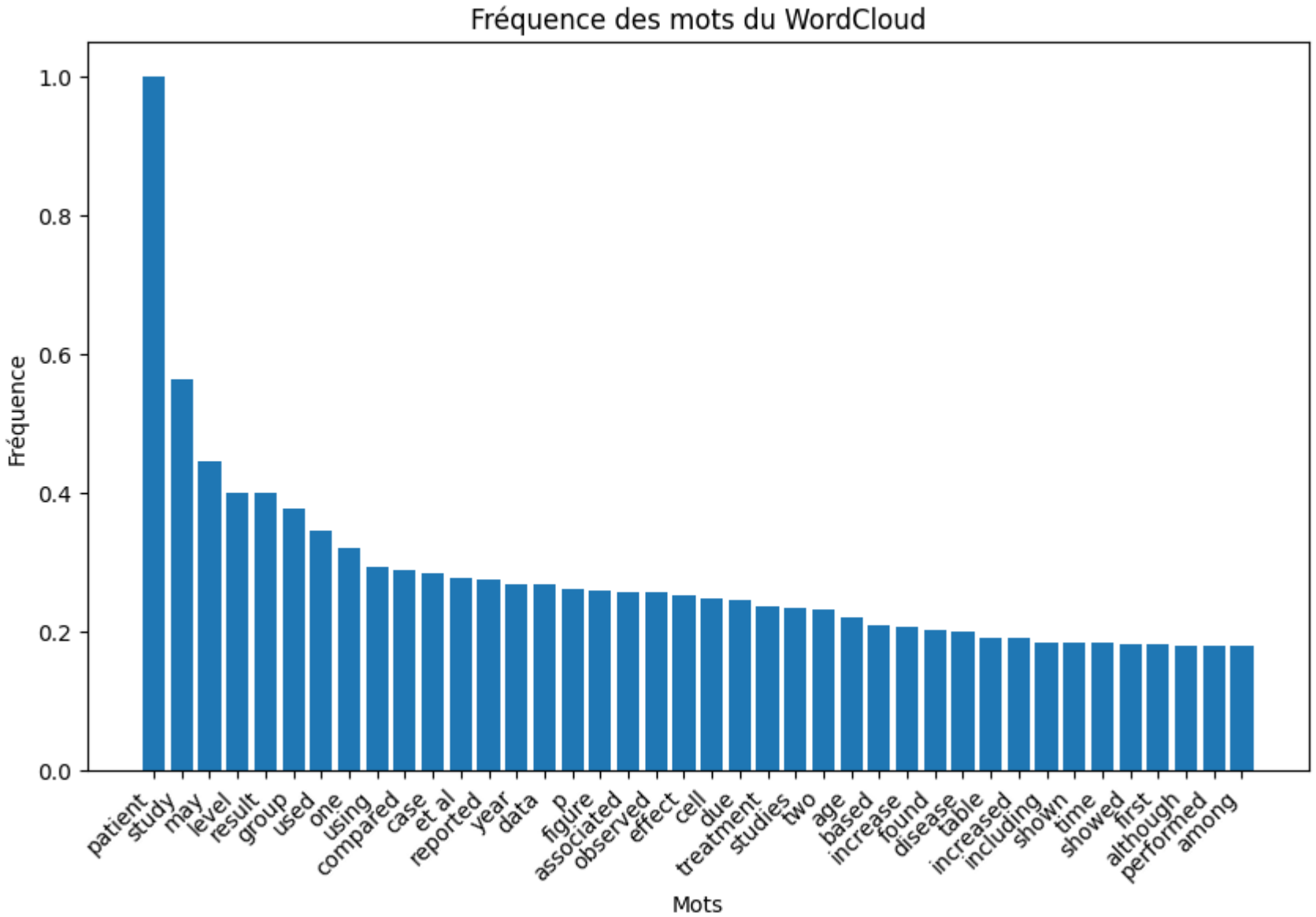
In [ ]:

```
def wordFrequency(wordCloud):
  keys = list(wordCloud.words_.keys())[:40]
  values = [wordCloud.words_[k] for k in keys]
```

```
plt.figure(figsize=(10, 6))
plt.bar(keys, values)
plt.title('Fréquence des mots du WordCloud')
plt.xlabel('Mots')
plt.ylabel('Fréquence')
plt.xticks(rotation=45, ha='right')
plt.show()
```

In [ ]:

```
wordFrequency(worldCloudTrain)
```



Après avoir créé cette fonction, on peut désormais voir les fréquences des mots dans le nuage de mots. Cependant, ce type de datavisualisation a des limites. Par exemple, comment sont calculées les fréquences par le WordCloud, et pourquoi voit-on des mots comme "p"?

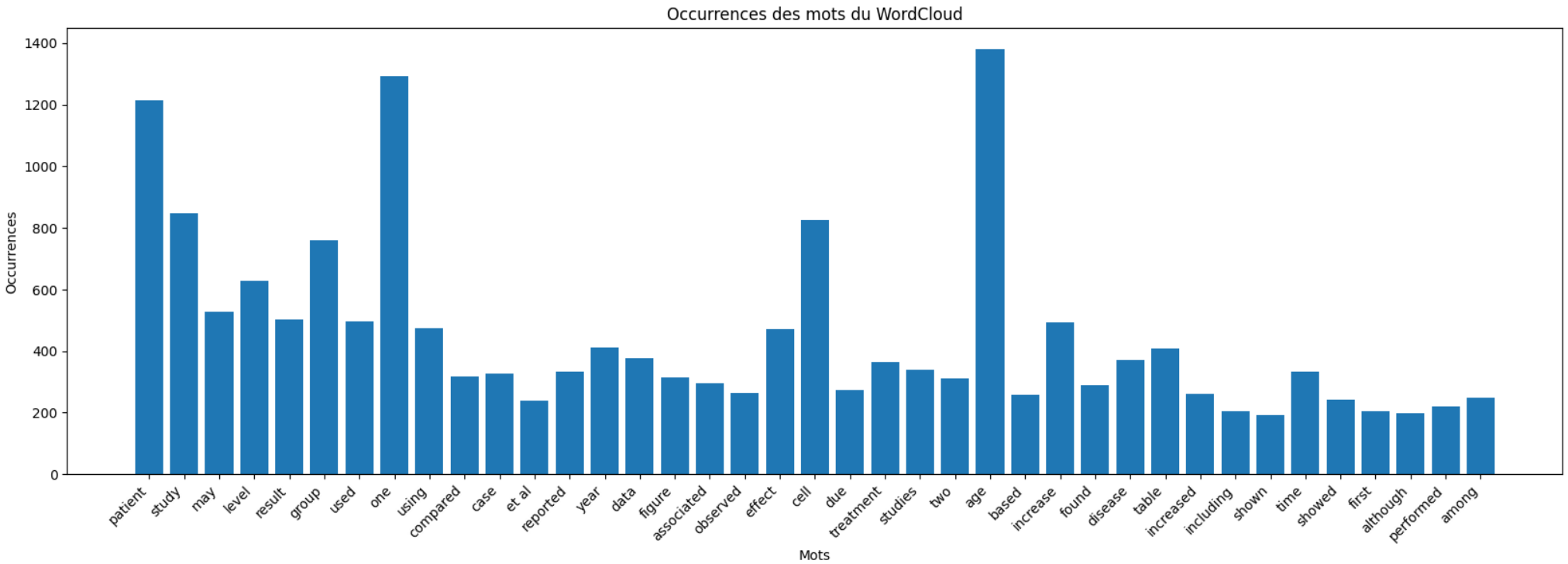
C'est donc à ce moment que notre deuxième fonction intervient: wordOccurrence, permettant de générer un graphe avec les mots les plus utilisés dans les différents articles et leur nombre d'occurrences.

Bien évidemment, les résultats affichés par ces fonction doivent être relativisés par rapport à la taille de l'échantillon que l'on a choisi précédemment.

In [ ]:
```
def findWord(text,word): return text.lower().count(word.lower())

def wordOccurrence(wordCloud, text):
  words = [w for w in list(wordCloud.words_.keys())[:40] if len(w) > 1]
  occurrences_map = {w: findWord(text, w) for w in words}
  keys = list(occurrences_map.keys())
  values = list(occurrences_map.values())
  plt.figure(figsize=(20, 6))
  plt.bar(keys, values)
  plt.title('Occurrences des mots du WordCloud')
  plt.xlabel('Mots')
  plt.ylabel('Occurrences')
  plt.xticks(rotation=45, ha='right')
  plt.show()
```

In [ ]:
```
wordOccurrence(worldCloudTrain,textTrain)
```


Occurrences des mots du WordCloud
Occurrences
0
200
400
600
800
1000
1200
1400
patient
study
may
level
result
group
used
one
using
compared
case
et al
reported
year
data
figure
associated
observed
effect
cell
due
treatment
studies
two
age
based
increase
found
disease
table
increased
including
shown
time
showed
first
although
performed
among
Mots