



Institut européen

des métiers de la **traduction** | IEMT

Université de Strasbourg

Web, corpus, traduction : exploitations

Prétraitements de données et visualisation

Enzo Doyen

2025 - M1

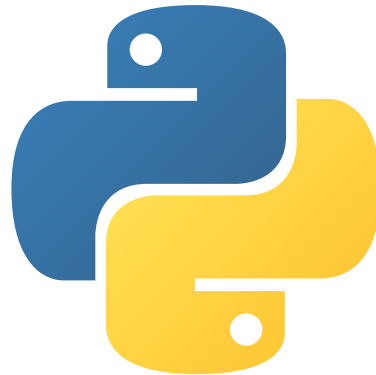
Prétraitements automatiques

Pour faciliter les analyses sur corpus et produire des graphiques intéressants sur des données textuelles, il est bien souvent nécessaire de faire du **prétraitement**.

Prétraitements automatiques

Pour faciliter les analyses sur corpus et produire des graphiques intéressants sur des données textuelles, il est bien souvent nécessaire de faire du **prétraitement**.

Ces **étapes de prétraitement** sont bien souvent réalisées **automatiquement** à l'aide de langages de programmation, le plus utilisé étant **Python**.



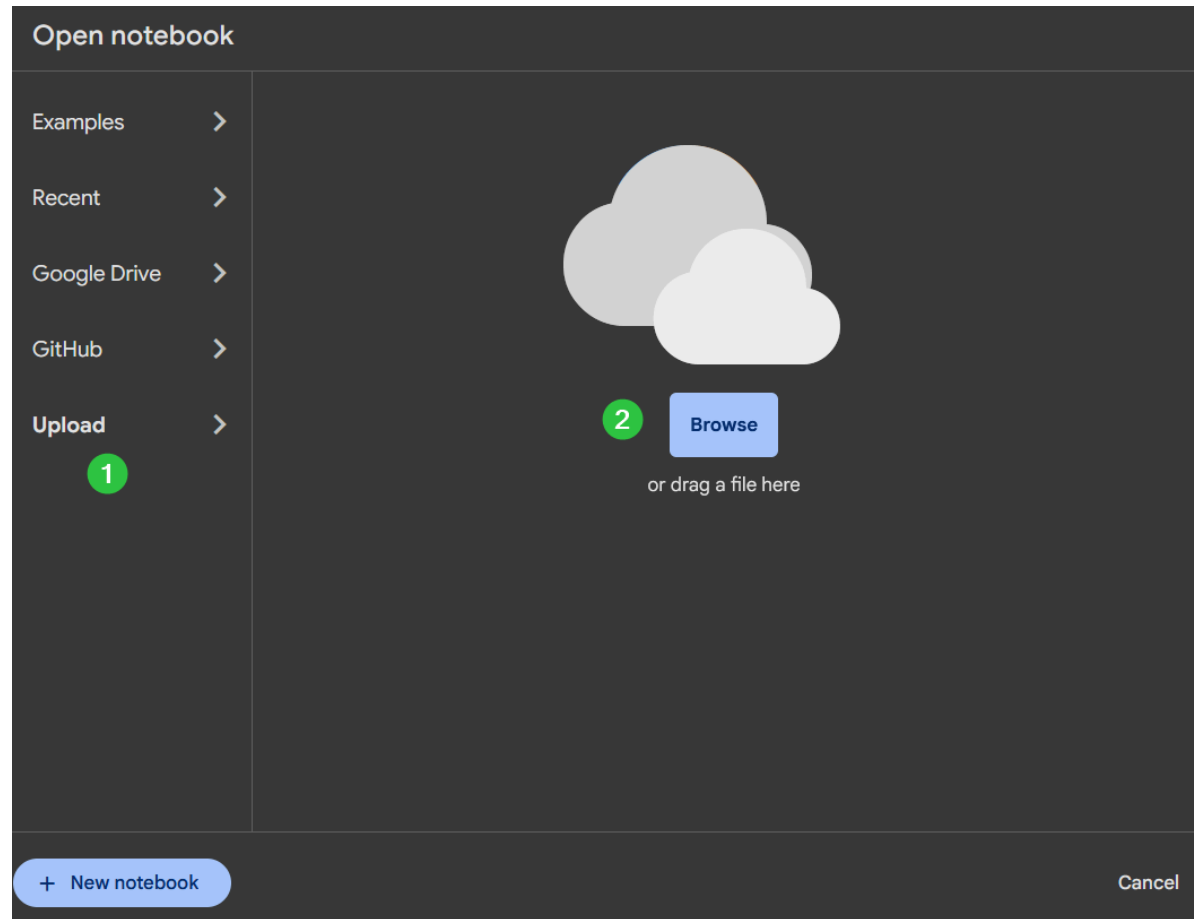
Google Colab

Avec des services en ligne tels que **Google Colab**, on peut facilement utiliser du code préexistant sous forme de « **notebook** » (un fichier au format `.ipynb`) et ainsi appliquer ces prétraitements sans grandes connaissances en programmation.

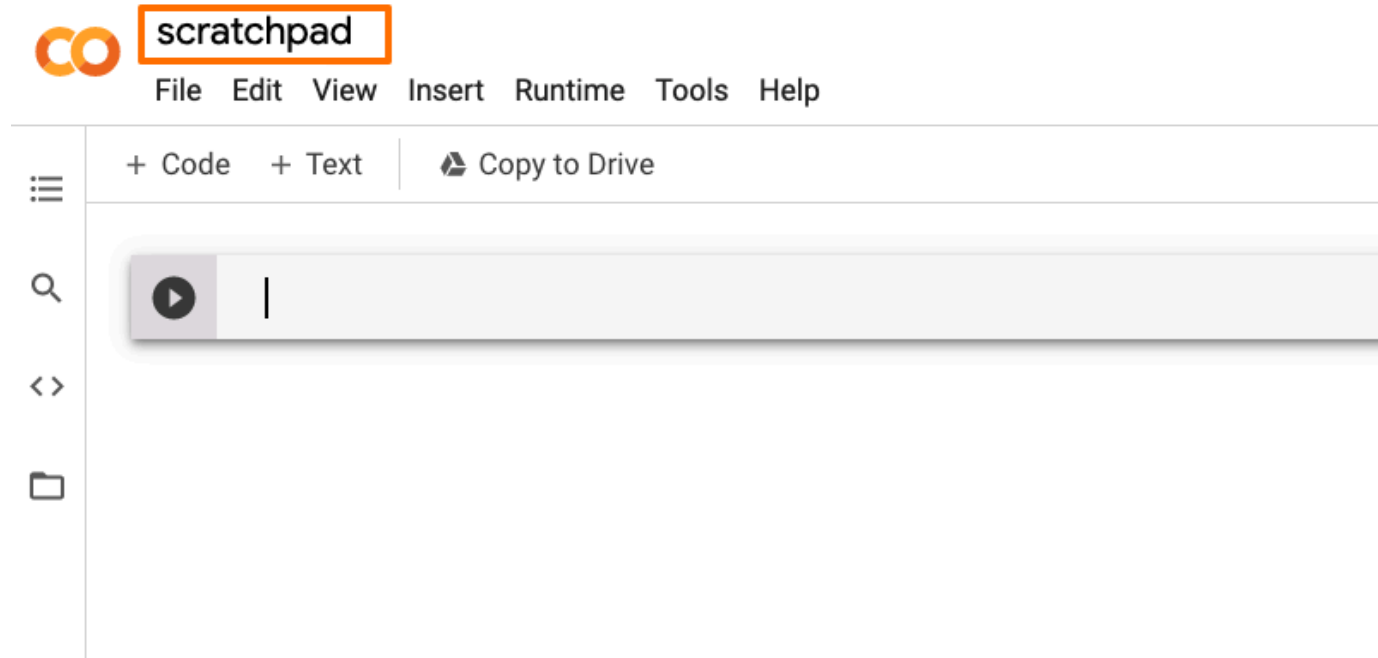


<https://colab.research.google.com/>

Google Colab : mise en ligne d'un notebook



Interface de Google Colab



Source : <https://www.analyticsvidhya.com/blog/2021/05/10-colab-tips-and-hacks-for-efficient-use-of-it/>

Google Colab : environnement d'exécution

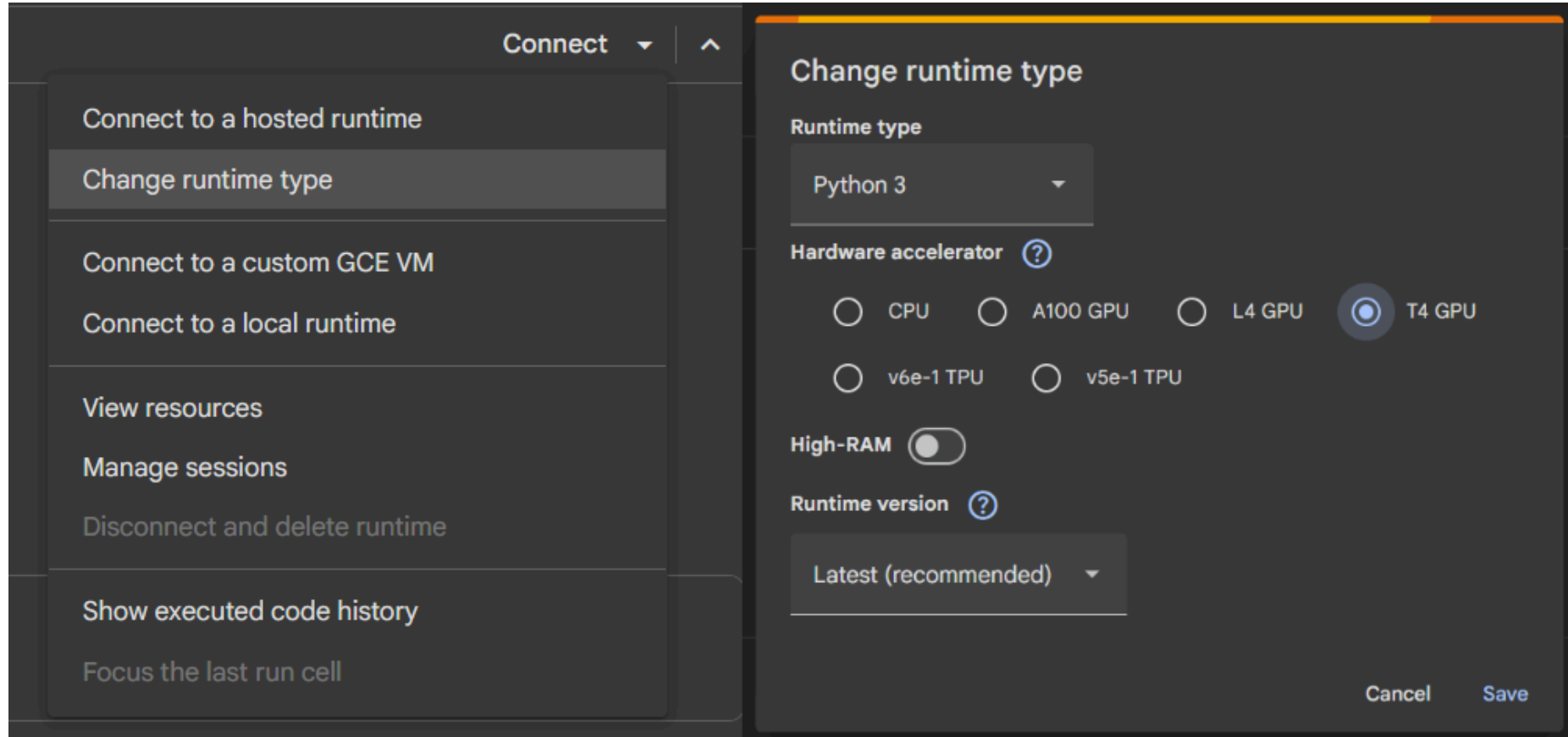
Google Colab fonctionne en vous connectant à un serveur dans le cloud hébergé par Google : cela signifie que les opérations de prétraitement ne sont pas appliquées sur votre ordinateur, mais sur une autre machine distante.

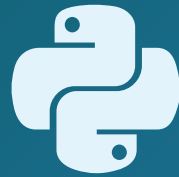
L'avantage est que cela permet de réaliser des opérations complexes sans utiliser les ressources de votre ordinateur, qui pourrait ne pas disposer d'une puissance de calcul suffisante.

Google Colab : environnement d'exécution

Il est possible de choisir le type de serveur auquel vous souhaitez vous connecter (plus ou moins puissant). Pour accélérer les opérations (notamment pour la visualisation avancée), il est recommandé de se connecter sur GPU.

Google Colab : environnement d'exécution GPU





I. Introduction élémentaire à Python

Python : opérateurs

Les opérateurs mathématiques de base sont pris en charge par Python :

1 5 + 9

2 5 / 2

3 (6 + 10) * 4

4 10 - 2



Python : commentaires

On peut ajouter des commentaires à notre code en utilisant un croisillon #.

```
1 5 + 9 # Résultat : 14
2 5 / 2 # Résultat : 2.5
3 (6 + 10) * 4 # Résultat : 64
4 10 - 2 # Résultat : 8
```



Python : impression avec print ()

print () est une **fonction** qui permet d'imprimer le résultat d'une opération.
Par exemple :

```
1 print(5 + 9)
```



Imprimera :

14

Python : affectation

On peut définir des **variables** pour sauvegarder le résultat d'une opération et le réutiliser plus tard. Pour ce faire, on indique le nom de notre variable suivi du signe égal = et de l'opération :

```
1 a = 5 + 9
```



On peut ensuite, par exemple, imprimer le résultat en réutilisant le même nom de variable :

```
1 print(a)
```

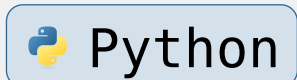


14

Python : chaînes de caractères

Le même type d'opération existe pour du texte (chaines de caractères, ou *strings* en anglais).

```
1 texte = "Bonjour !"
2 print(texte)
```



Bonjour !

Python : exemple dans Google Colab

Voici un exemple d'affectation d'une chaîne de caractère à une variable dans Google Colab :

```
[ ]
```

```
1 # Indiquer le nom du fichier complet mis en ligne dans Google Colab ici  
2 CORPUS_PATH = "ecologie_pluri.txt"
```

Python : fonctions

Les **fonctions** sont utilisées pour regrouper plusieurs opérations à effectuer à la suite sous forme de « blocs logiques ».

Elles permettent notamment de transmettre des valeurs à utiliser pour les opérations à l'intérieur (des **paramètres**).

```
1 def greet(name):  
2     print("Bonjour " + name + " !")  
3  
4 greet("Léa")
```



Bonjour Léa !

Python : conditions

```
1 def greet(name, is_night):  
2     if is_night is True: # ou simplement "if is_night"  
3         print("Bonsoir " + name + " !")  
4     elif is_night is False: # ou simplement "else:"  
5         print("Bonjour " + name + " !")  
6  
7 greet("Léa", True)
```



Bonsoir Léa !

Python : bibliothèques externes

Il est possible d'utiliser des variables, fonctions et autres types d'opérations déjà créées en **important** des **bibliothèques** avec le mot-clé `import` :

```
1 import spacy
2 import re
3 import string
4 import sys
5 from spacy.lang.fr.stop_words import STOP_WORDS
6 from unicodedata import category
```



Python : bibliothèques externes

Il est possible d'utiliser des variables, fonctions et autres types d'opérations déjà créées en **important** des **bibliothèques** avec le mot-clé `import` :

```
1 import spacy
```

```
2 import re
```

```
3 import string
```

```
4 import sys
```

```
5 from spacy.lang.fr.stop_words import STOP_WORDS
```

```
6 from unicodedata import category
```



Par exemple, la bibliothèque (ou module) `re` est utilisée pour les expressions régulières.



II. Nettoyage de corpus

Nettoyage de corpus

Grâce au langage Python présenté précédemment, on peut très facilement appliquer plusieurs étapes de nettoyage sur notre corpus pour, par exemple :

- ♦ supprimer les mots grammaticaux ou sémantiquement pauvres (« mots vides ») qui ajoutent du bruit dans nos données ;
- ♦ nettoyer des fichiers exportés depuis certains services (p. ex. SketchEngine ou Europresse) qui modifient les textes initiaux pour les rendre compatibles avec leur système ;
- ♦ corriger les problèmes de formatage dans nos textes (p. ex., espaces en trop) ;
- ♦ supprimer certains éléments peu utiles, comme des noms propres.

Suppression des mots vides (*stopwords*)

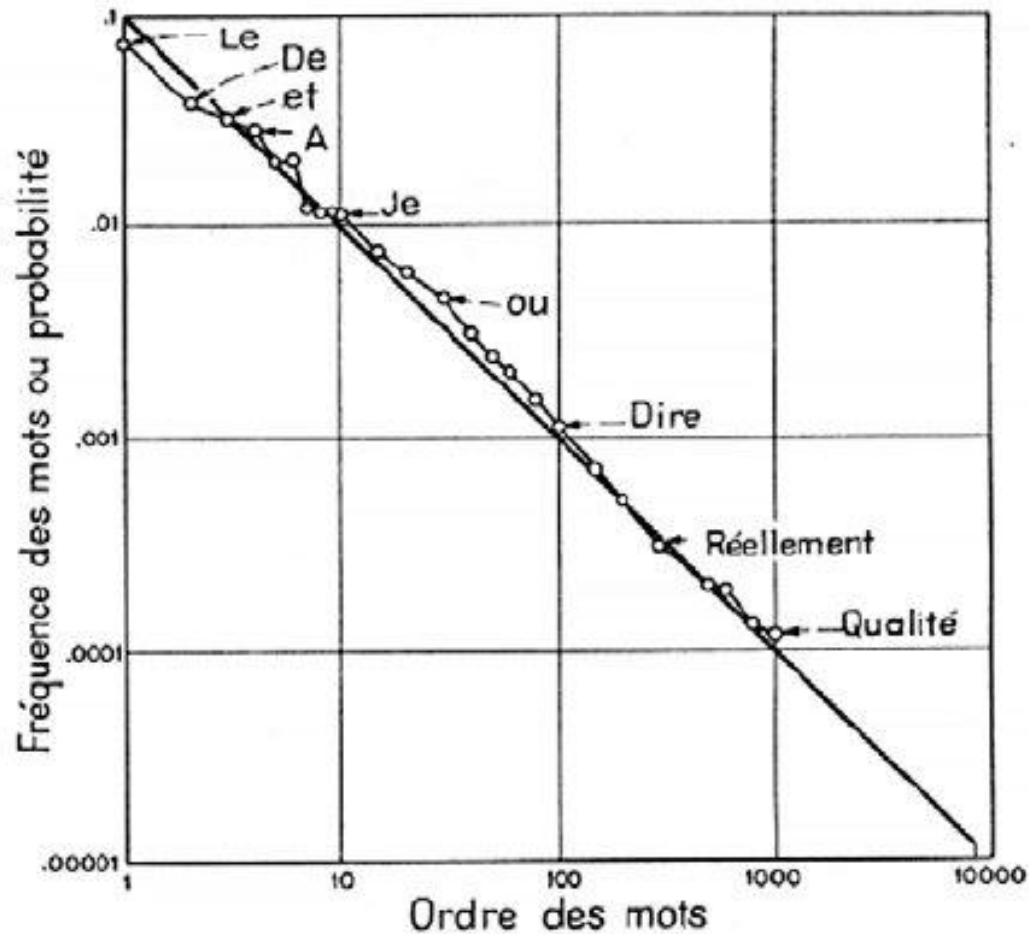
Les mots vides sont des mots qui n'apportent pas de valeur sémantique à un document, et qui peuvent être supprimés pour donner davantage d'importance à des mots sémantiquement forts.

Ils incluent bien souvent les mots grammaticaux, par exemple en français :
« le », « la », « et », « à », « de », etc.

Suppression des mots vides (*stopwords*)

Lien avec la **loi de Zipf** (Zipf, 1932) : observation empirique selon laquelle, dans un corpus de textes, la fréquence d'un mot est inversement proportionnelle à son rang dans la liste des mots par fréquence.

Un mot de rang 1 (le plus fréquent) apparaîtra deux fois plus souvent qu'un mot de rang 2, trois fois plus souvent qu'un mot de rang 3, etc.



Représentation graphique de la loi de Zipf. Source : **Mandelbrot (1965)**

Suppression des mots vides (*stopwords*) avec *spaCy*

spaCy est une bibliothèque Python utilisée pour le traitement automatique des langues, mais on peut aussi l'utiliser simplement pour filtrer les mots vides.

Il suffit d'importer la variable `STOP_WORDS`, qui contient une liste prédéfinie de mots vides pour le français :

```
1 from spacy.lang.fr.stop_words import  
  STOP_WORDS
```



Suppression des mots vides (*stopwords*) avec *spaCy*

```
STOP_WORDS = set(  
    """"  
    a à â abord afin ah ai aie ainsi ait allaient allons  
    alors anterieur anterieure anterieures antérieur antérieure antérieures  
    apres après as assez attendu au  
    aupres auquel aura auraient aurait auront  
    aussi autre autrement autres autrui aux auxquelles auxquels avaient  
    avais avait avant avec avoir avons ayant  
  
    bas basee bat
```

Ajouter des mots à la liste de mots vides

Pour ajouter des mots supplémentaires à la liste initiale de spaCy, on peut utiliser la **méthode** `add` (un type de fonction) de cette manière :

```
1 # Ajoute le mot « éléphant » à la liste de  
   mots vides
```



```
   # Cette ligne doit être exécutée dans le notebook  
2 *avant* de réaliser les étapes de prétraitement et de  
   visualisation !  
3 STOP_WORDS.add("éléphant")
```

Voir si un mot est dans la liste de mots vides

Il est également possible de voir si un mot spécifique se trouve dans la liste de mots vides de spaCy avec l'opérateur `in`. Le résultat renvoyé est `True` (« vrai ») si c'est le cas, `False` (« faux ») sinon.

```
1 "mais" in STOP_WORDS
```



True

```
1 "nounours" in STOP_WORDS
```



False

Université de Strasbourg



Université de Strasbourg



Suppression des balises de SketchEngine

```
<doc id="file33628896" filename="10-mesures-  
developpement-maitrise-responsable-leolien.html"  
1 parent_folder="web1" url="https://www.ecologie.gouv.  
fr/actualites/10-mesures-developpement-maitrise-  
responsable-leolien">  
2 <p> 10 mesures pour un développement maîtrisé et  
responsable de l'éolien </p>  
3 <p> Comment assurer un développement maîtrisé et  
responsable de l'éolien ? </p>
```


Suppression des balises de SketchEngine

```
1 def clean_sketchengine_tags(text):  
2     cleaned_text = re.sub(r'<[^>]+>', '', text)  
3     return cleaned_text
```



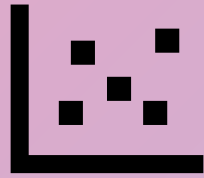
Suppression des balises de SketchEngine

```
1 def clean_sketchengine_tags(text):  
2     cleaned_text = re.sub(r'<[^>]+>', '', text)  
3     return cleaned_text
```

 Python

Résultat :

- 1 10 mesures pour un développement maîtrisé et responsable de l'éolien
- 2 Comment assurer un développement maîtrisé et responsable de l'éolien ?



III. Visualisation

Bibliothèques de visualisation

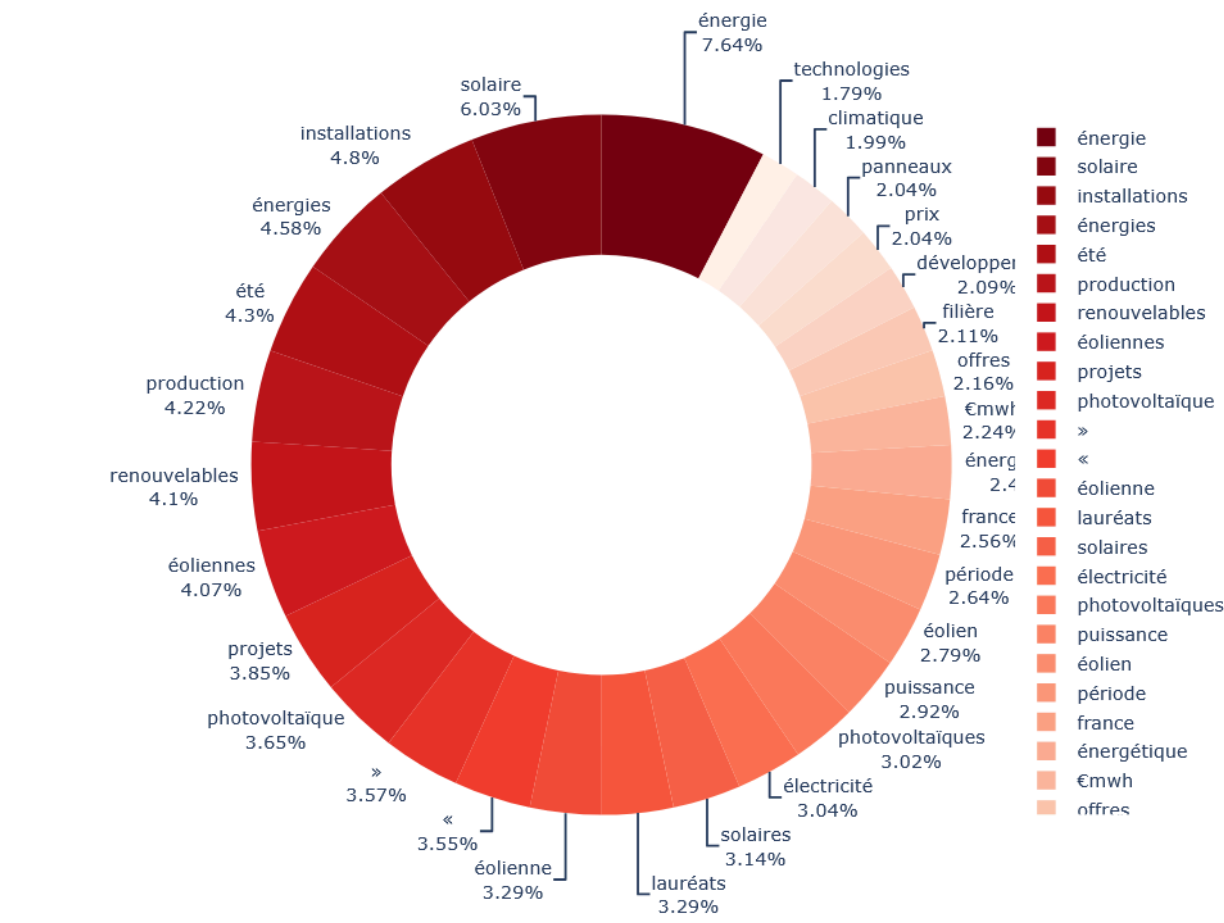
Il existe de nombreuses bibliothèques Python qui permettent de créer facilement des visualisations (simples ou avancées) de données textuelles (ou autres types de données).

Parmi elles : matplotlib, seaborn, plotly, circlify [🔗], ou scattertext [🔗] et wordcloud [🔗] (spécifiquement pour les données textuelles).

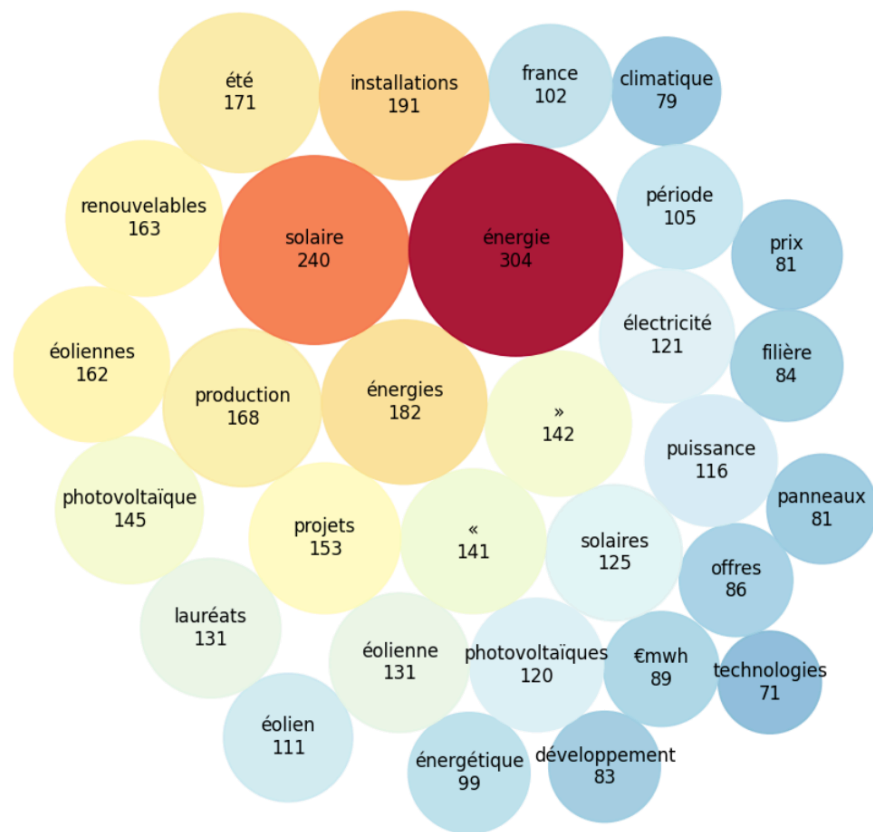
Université de Strasbourg



Visualisation simple : graphique « donut »



Visualisation simple : graphique « cercles »



Visualisation avancée : *scattertext*

La bibliothèque *scattertext* produit un graphique interactif au format HTML (le même format qu'une page Web) en comparant votre corpus avec un corpus de référence.

Vous pouvez ainsi voir les termes les plus spécifiques de votre corpus sous forme graphique, et voir aussi les contextes d'utilisation.

Visualisation avancée : *scattertext*

La bibliothèque *scattertext* produit un graphique interactif au format HTML (le même format qu'une page Web) en comparant votre corpus avec un corpus de référence.

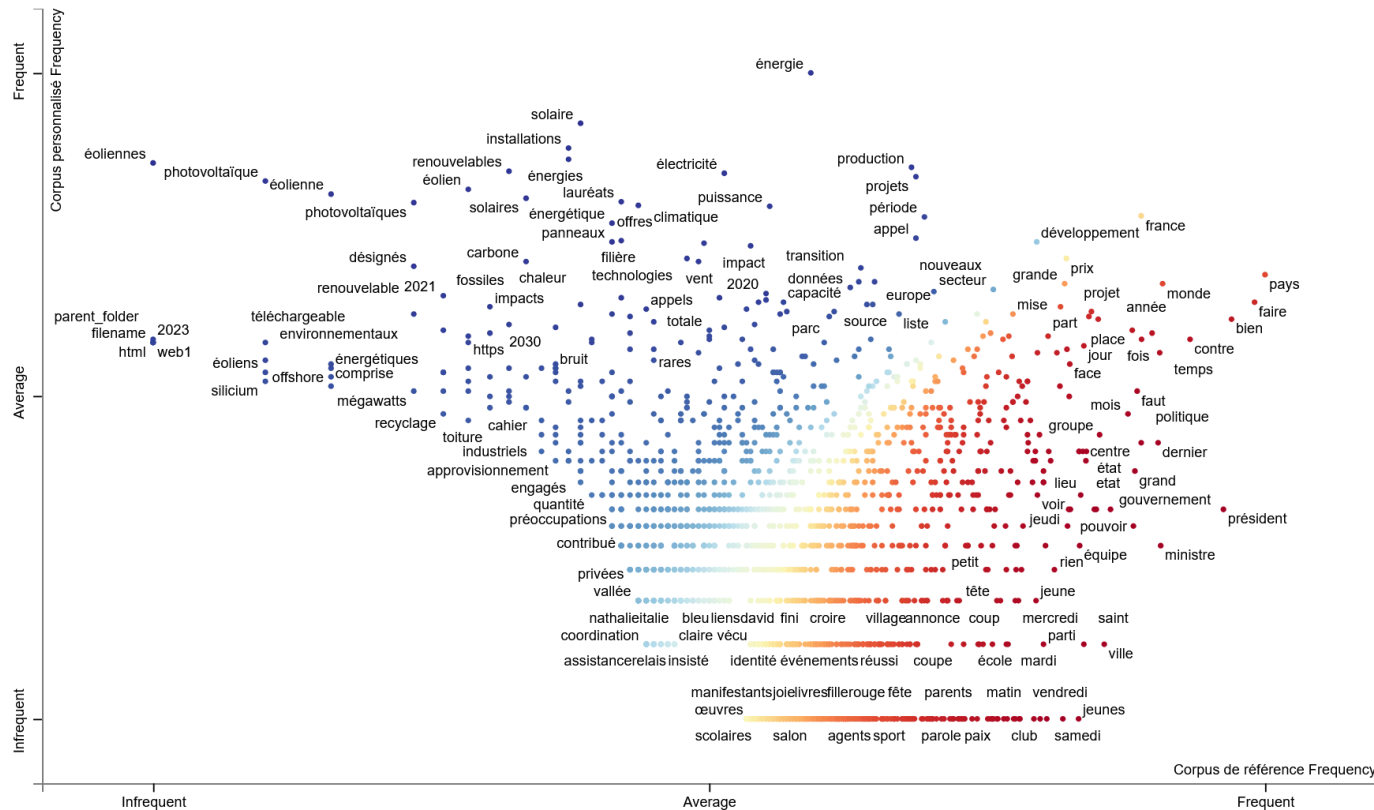
Vous pouvez ainsi voir les termes les plus spécifiques de votre corpus sous forme graphique, et voir aussi les contextes d'utilisation.



Attention

scattertext s'utilise avec un corpus **non** prétraité, car la bibliothèque applique elle-même ses propres étapes de prétraitement.

Visualisation avancée : *scattertext*



Corpus personnalisé document count: 1,570; word count: 57,921
Corpus de référence document count: 26,646; word count: 573,472

Search the chart

Top Corpus personnalisé

éoliennes
solaire
installations
énergies
renouvelables
photovoltaïque
éolien
éolienne
solaire
photovoltaïques
électricité
lauréats
énergie
offres

Characteris

aujourd
renouvelable
dernier
solaire
ministre
explique
solaire
ailleurs
faut
pouvoir
prochain
agit
autant
veut
milliards

Top Corpus de référence

ministre
ville
président
gouvernement
saint
jeunes
etat
nationale
pouvoir
grand
chef
équipe
dernier
politique
fois
gouverneme
peuple
contre
devrait
beaucoup
mois
panneaux
prendre
vient
samedi
vendredi
toutefois
climatique
citoyens

Visualisation avancée : *scattertext*

Comparaison des contextes avec un corpus de référence

Term: **éolien**

Corpus personnalisé frequency:
193 per 25,000 terms
68 per 1,000 docs
The 136 mentions in 107 available documents:

10 mesures pour un développement maîtrisé et responsable de l'**éolien**

Comment assurer un développement maîtrisé et responsable de l'**éolien** ?

L'objectif est d'activer « tous les leviers en même temps », donc de poursuivre le déploiement de l'**éolien**, pour assurer à long terme la sécurité d'approvisionnement et être à la hauteur des ambitions climatiques de la France, a-t-elle poursuivi.

« L'**éolien** est une énergie renouvelable, décarbonée et bon marché qui a toute sa place dans notre mix énergétique.

Nouvelles mesures en faveur d'un **éolien** responsable

C'est le sens des 10 nouvelles mesures annoncées aujourd'hui, en lien avec la filière, pour aller vers un **éolien** plus responsable.

Corpus de référence frequency:
1 per 25,000 terms
0 per 1,000 docs
The 6 mentions in 6 available documents:

La préfète de Normandie, Nicole Klein, a signé le mois dernier l'autorisation permettant à la société Éolienne offshore des Hautes-Falaises de construire et d'exploiter le parc **éolien** en mer au large de Fécamp.

Mais selon les projections, en 2025, la production d'énergies renouvelables atteindrait 115 gigawatts (GW), ou plus précisément : 65 GW pour l'**éolien**, 24 GW pour le solaire et 26 GW pour l'hydroélectrique.

Ordre du jour: projet **éolien**, débatset vote.

Pour les autorités, "le peuple s'est déjà prononcé pour le projet de parc **éolien** et il a validé la démarche".

25684 Quant à l'**éolien**, ça va vite finir.

WEILER – Mercredi, un milan royal a été retrouvé mort dans le parc **éolien** OekoStroum, situé à Weiler.

Outils de visualisation en ligne

Il existe également des outils de visualisation en ligne, sous forme d'interface, comme par exemple **Voyant Tools**.



IV. Mise en pratique

Mise en pratique

Sur Moodle est mis à disposition un notebook (fichier .ipynb) que vous pouvez mettre en ligne dans Google Colab (voir diapositive n° 3) pour effectuer les prétraitements sur votre corpus et générer des visualisations de ce dernier.

Bibliographie

Mandelbrot, B. B. (1965). Information Theory and Psycholinguistics. In B. B. Wolman et E. Nagel (éds.), *Scientific Psychology: Scientific Psychology*. Basic Books.

Zipf, G. K. (1932). *Selected Studies of the Principle of Relative Frequency in Language* (p. 57). Harvard Univ. Press. [10.4159/harvard.9780674434929](#)