Université de Strasbourg

Recherche d'information

Prétraitements et annotation des données textuelles

Enzo Doyen

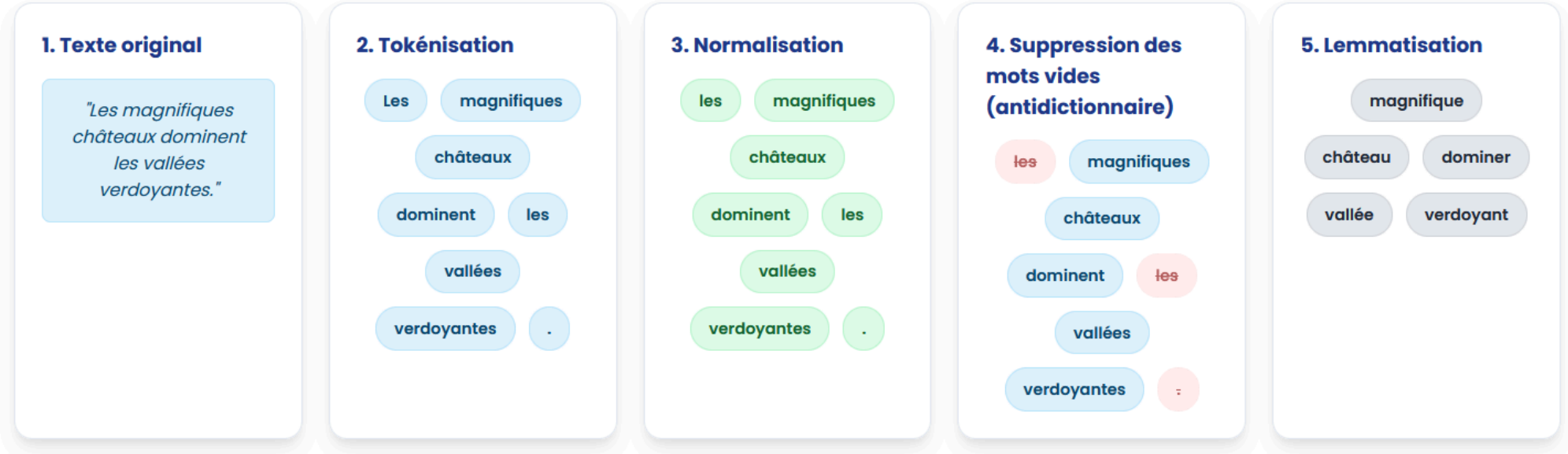
2025 - LGC6KM43 - M2

Prétraitement des données textuelles

Avant d'indexer des documents pour de la recherche d'information, et pour améliorer l'efficacité de la recherche, il est souvent nécessaire de les traiter au prélable (en d'autres termes, opérer des « prétraitements »).

L'objectif est de passer d'un **texte brut** à un **texte lexicalement et sémantiquement riche**.

Prétraitement des données textuelles



Étapes de prétraitement des données textuelles : tokénisation, normalisation, suppression des mots vides (*stopwords*) et lemmatisation

I. Méthodes de prétraitement

Tokénisation

Original	Tokénisation simple	Tokenisation avancée
l'école	["l", "'", "école"]	["l'", "école"]
n'a	["n", "'", "a"]	["n'", "a"]
aujourd'hui	["aujourd", "'", "hui"]	["aujourd'hui"]
arrière-grand-père	["arrière", "-", "grand", "-", "père"]	["arrière-grand-père"]
Bourg-en-Bresse	["Bourg", "-", "en", "-", "Bresse"]	["Bourg-en-Bresse"]
jean.d@email.fr	["jean", ".", "d", "@", "email", ".", "fr"]	["jean.d@email.fr"]
12€50	["12", "€", "50"]	["12€50"]

Tokénisation avec *NLTK*

```python
import nltk
from nltk.tokenize import word_tokenize
nltk.download("punkt")
text = "Les chiens ont l'habitude d'aboyer tous les matins."
tokens = word_tokenize(text, language="french")
```

Sortie : `['Les', 'chiens', 'ont', "l'habitude", 'd', 'aboyer', 'tous', 'les', 'matins', '.']`

Tokénisation avec *spaCy*

Python

```python
import spacy

nlp = spacy.load("fr_core_news_sm")
text = "Les chiens ont l'habitude d'aboyer tous les matins."
doc = nlp(text)
tokens = [token.text for token in doc]
```

spaCy et modèles pour le français

Dans *spaCy*, modèles de différentes tailles disponibles pour le français, avec différents corpus d'entrainement et différentes architectures.

Un modèle plus grand donne généralement de meilleures performances, mais est aussi plus lourd et plus lent à charger/utiliser.

---

https://spacy.io/models/fr

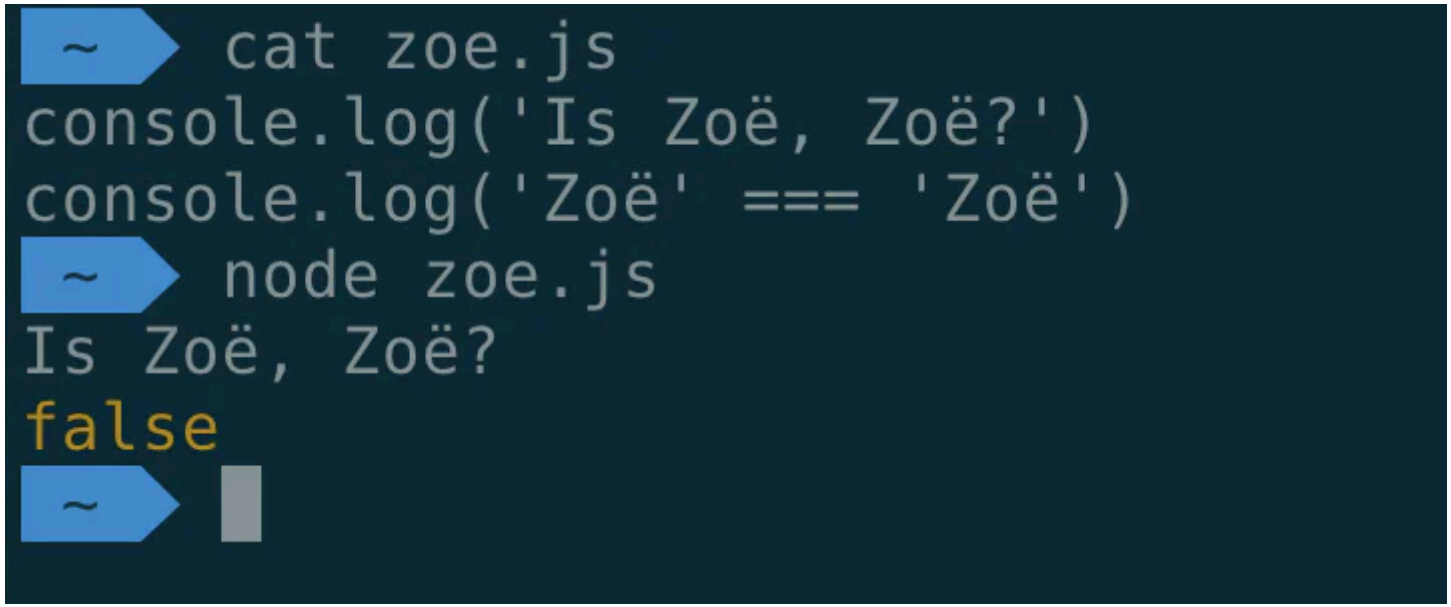
Modèles disponibles : fr_core_news_sm (petit), fr_core_news_md (moyen), fr_core_news_lg (grand), fr_core_news_trf (Transformer)

Normalisation

La **normalisation** consiste à standardiser le texte en éliminant les variations, pour ainsi faciliter le traitement ultérieur.

Normalisation

La **normalisation** consiste à standardiser le texte en éliminant les variations, pour ainsi faciliter le traitement ultérieur.



Source : https://withblue.ink/2019/03/11/why-you-need-to-normalize-unicode-strings.html

Normalisation

La **normalisation** consiste à standardiser le texte en éliminant les variations, pour ainsi faciliter le traitement ultérieur.

Par exemple, une normalisation voulue en français pour le mot « œuf » :

`œuf = oeuf = Œuf = OEUF = ŒUF`

Normalisation : casse

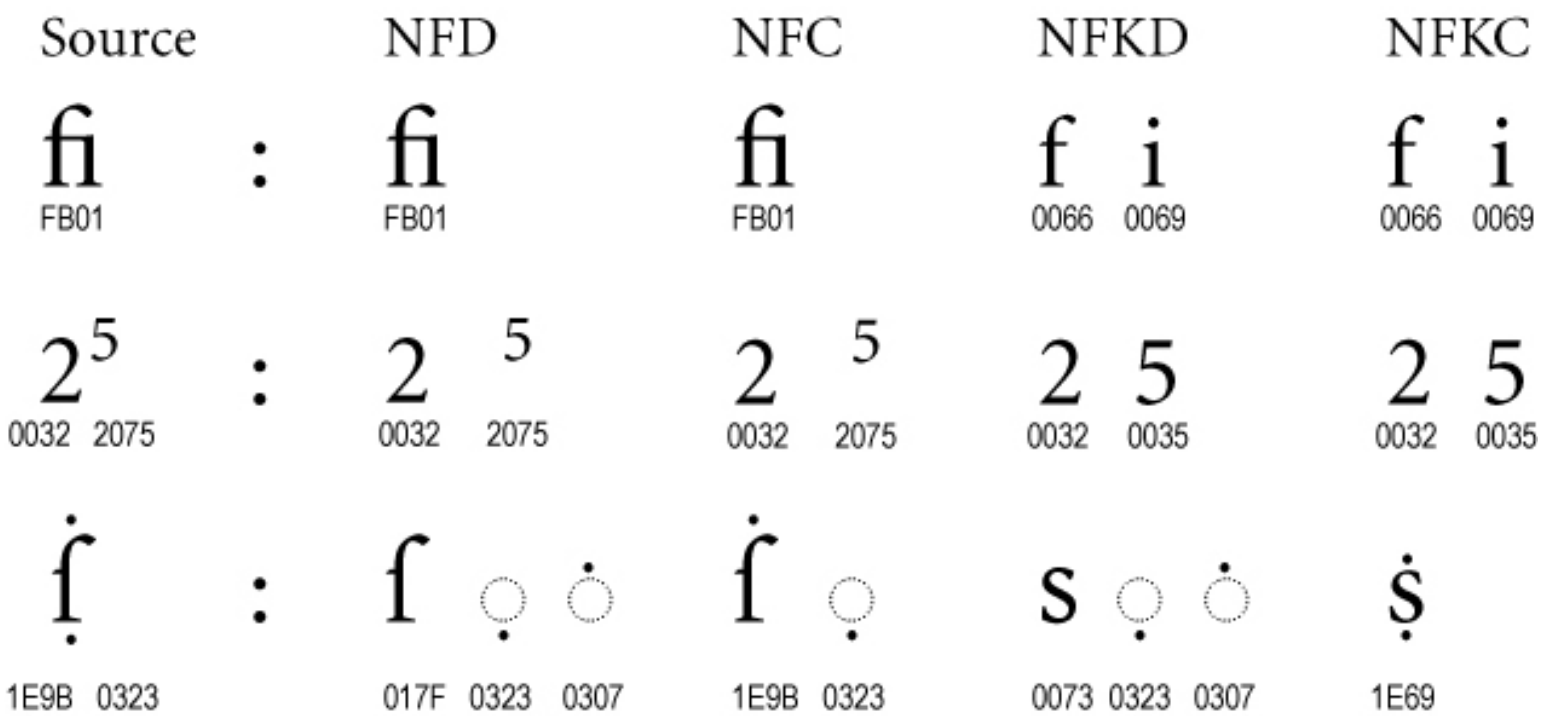
La normalisation au niveau de la casse se fait généralement en convertissant tout le texte en minuscules :

```python
_ = "OEUF".lower()
```

Normalisation : espaces

```python
_ = " oeuf ".strip()
_ = " oeuf".lstrip()
_ = "oeuf ".rstrip()
_ = " ".join(" un  oeuf sur  la    table".split())
```

Normalisation : diacritiques et ligatures



Source : **https://www.unicode.org/reports/tr15/**

Normalisation : diacritiques et ligatures

```python
import unicodedata
unicodedata.normalize('NFKC', "Ĳ") # 'IJ'
unicodedata.normalize('NFKD', "½") # '1/2'
```

```python
from unidecode import unidecode
unidecode("épée", "utf-8") # 'epee'
```

> ⚠ **Attention**
>
> Certaines ligatures, considérées comme des lettres à part entières, ne sont pas décomposables directement (p. ex. « œ », « æ »).

Suppression de la ponctuation

Python

```python
import string

text = text.translate(str.maketrans("", "", string.punctuation))
```

Suppression des mots vides (*stopwords*)

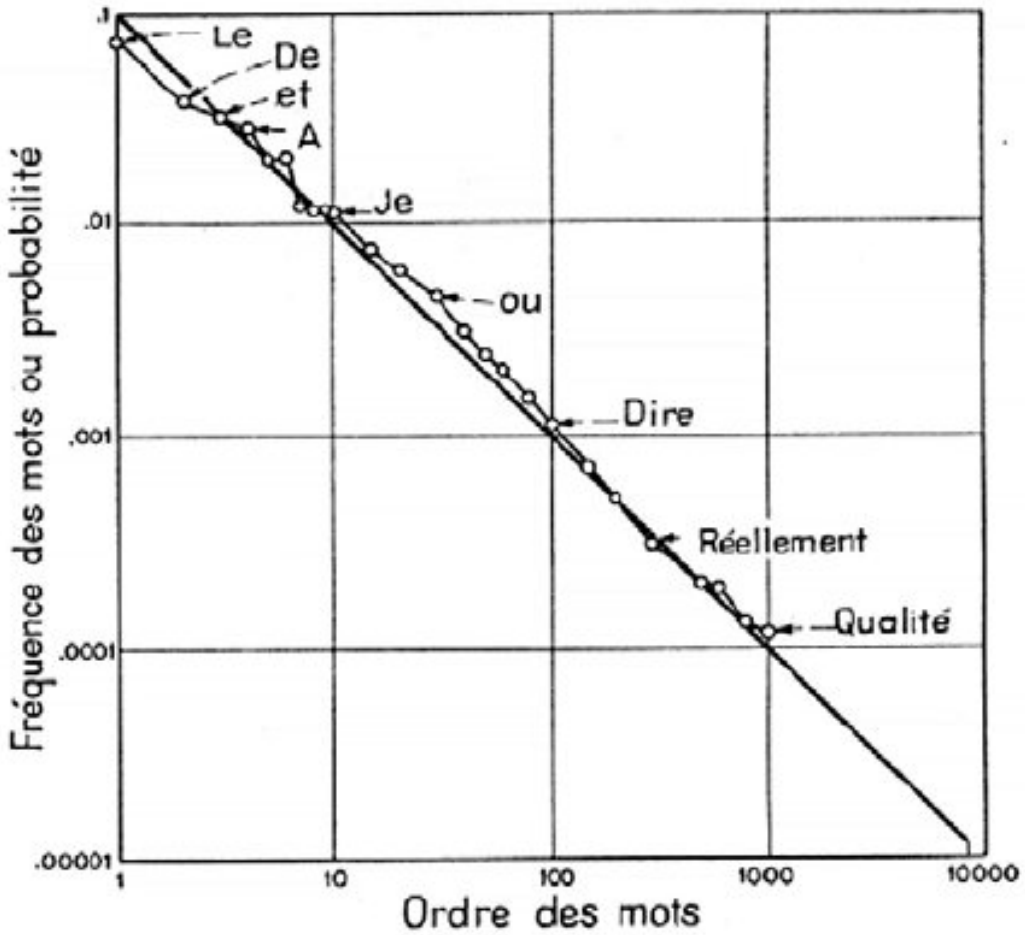
Les mots vides sont des mots qui n'apportent pas de valeur sémantique à un document, et qui peuvent être supprimés pour alléger l'indexation.

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 texte, 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) apparaitra 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 *NLTK*

Python

```python
import nltk
from nltk.corpus import stopwords

nltk.download("stopwords")
english_sw = stopwords.words('english')
french_sw = stopwords.words('french')
```

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

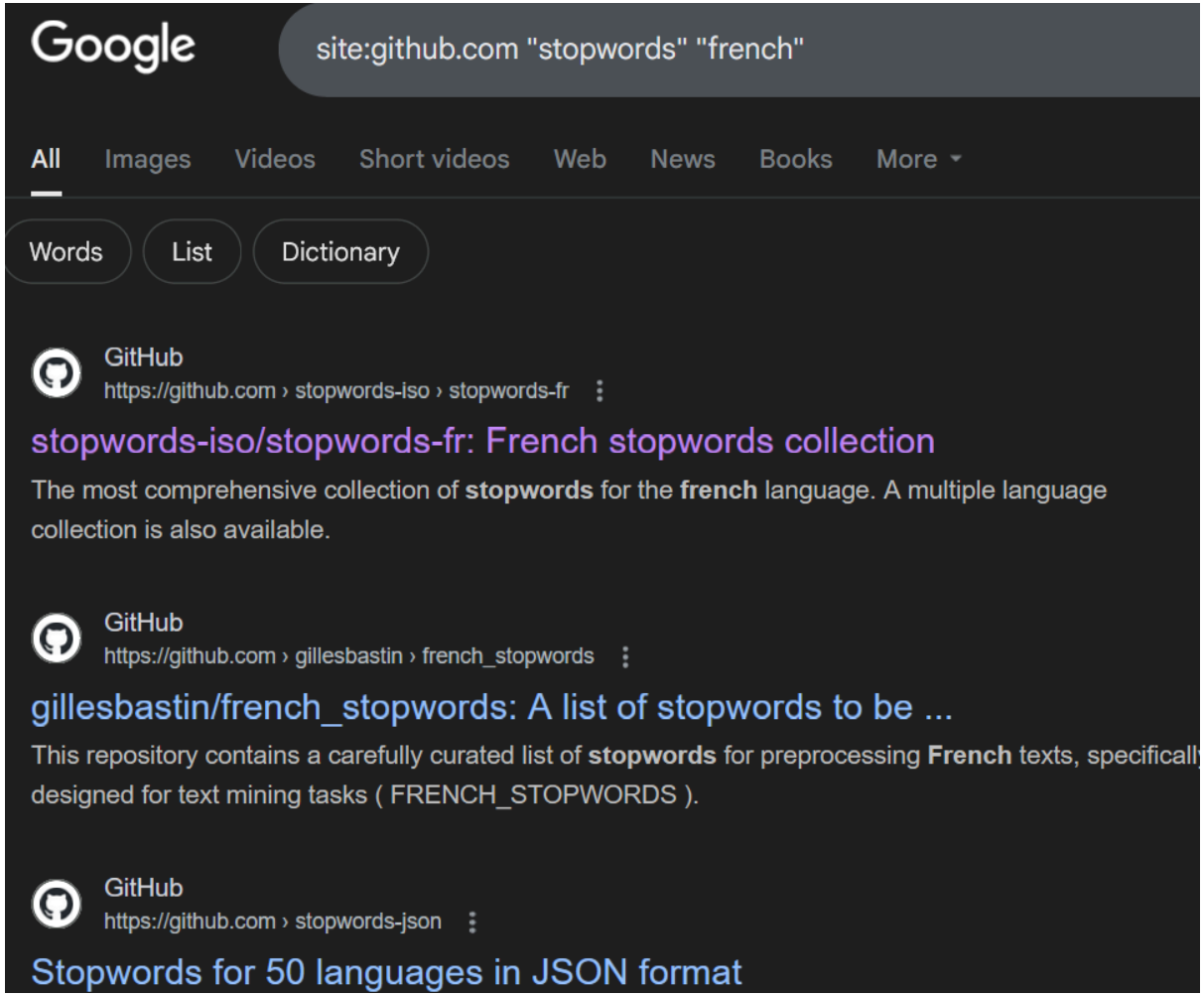
Python

```python
import spacy
from spacy.lang.en import stop_words

nlp = spacy.load('en_core_web_sm')

stop_words = stop_words.STOP_WORDS
```

Suppression des mots vides (*stopwords*)

Liste en français pour spaCy : https://github.com/explosion/spaCy/blob/master/spacy/lang/fr/stop_words.py

```
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
```


Google
site:github.com "stopwords" "french"
All
Images
Videos
Short videos
Web
News
Books
More
Words
List
Dictionary
GitHub
https://github.com › stopwords-iso › stopwords-fr
stopwords-iso/stopwords-fr: French stopwords collection
The most comprehensive collection of stopwords for the french language. A multiple language collection is also available.
GitHub
https://github.com › gillesbastin › french_stopwords
gillesbastin/french_stopwords: A list of stopwords to be ...
This repository contains a carefully curated list of stopwords for preprocessing French texts, specifically designed for text mining tasks (FRENCH_STOPWORDS).
GitHub
https://github.com › stopwords-json
Stopwords for 50 languages in JSON format

Lemmatisation

La lemmatisation est le processus de réduction des mots à leur forme canonique (leur « lemme »), ce qui permet de regrouper les différentes formes d'un mot.

Lemmatisation avec *spaCy*

Python

```python
import spacy

nlp = spacy.load("fr_core_news_sm")
text = "Les chiens ont l'habitude d'aboyer tous les matins."
lemmas = [token.lemma_ for token in nlp(text)]
```

Sortie : `['le', 'chien', 'avoir', 'le', 'habitude', 'de', 'aboyer', 'tout', 'le', 'matin', '.']`

Lemmatisation avec *treetaggerwrapper*

Python

```python
import treetaggerwrapper

tagger = treetaggerwrapper.TreeTagger(TAGLANG='fr')
tags = treetaggerwrapper.make_tags(tagger.tag_text(text))

lemmas = [t.lemma for t in tags if t.lemma is not None]
```

Autres types de prétraitements

- corrections de fautes d'orthographe ;
- correction phonétique ;
- *chunking* (découpage en phrases ou en segments, notamment pour les systèmes de *RAG*) ;
- suppression d'informations sensibles (par exemple, pour des raisons de confidentialité) ;
- …

II. Annotation automatique

Annotation automatique

Il est possible d'utiliser des techniques du domaine de l'**extraction d'informations** pour annoter automatiquement des documents avec des informations complémentaires.

Ces techniques peuvent permettre de récupérer des mots-clés du document (extraction de mots-clés, *keyword extraction*), des entités nommées (reconnaissance d'entités nommées, *named entity recognition*, NER), des relations entre entités (extraction des relations, *relationship extraction*), etc.

Les informations additionnelles obtenues grâce à ces techniques peuvent être utilisées pour améliorer la recherche et la visibilité des résultats, ou bien pour filtrer/regrouper des documents, par exemple.

Annotation automatique : extraction de mots-clés

Exploitation de différentes caractéristiques linguistiques pour trouver les mots les plus pertinents dans le texte : casse, position, fréquence au sein du document, fréquence au niveau des phrases, etc.

Exemples d'implémentation en Python :

- Yake (**Campos et al. (2018)** ; notebook disponible sur Moodle)
- pke (**Boudin (2016)** ; notebook disponible sur Moodle)

Annotation automatique : extraction de mots-clés

```python
import yake
kw_extractor = yake.KeywordExtractor()
keywords = kw_extractor.extract_keywords(text)
```

Python

```python
('google', 0.026580863364597897) # score plus bas = mot plus pertinent
('kaggle', 0.0289005976239829)
('ceo anthony goldbloom', 0.029946071606210194)
('san francisco', 0.04881083707482533 6)
```

Python

Annotation automatique : extraction de mots-clés

Utilisation de plongements lexicaux pour de meilleures performances, par exemple **KeyBERT** (basé sur sentence-transformers) avec un modèle multilingue **[?]**.

Annotation automatique : extraction de mots-clés

```python
from keybert import KeyBERT
kw_model = KeyBERT()
keywords = kw_model.extract_keywords(doc)
```

Python

```python
[('learning', 0.4604), # score plus haut = mot plus pertinent
('model', 0.4588),
('algorithm', 0.4556),
('training', 0.4487)]
```

Python

Annotation automatique : extraction de mots-clés

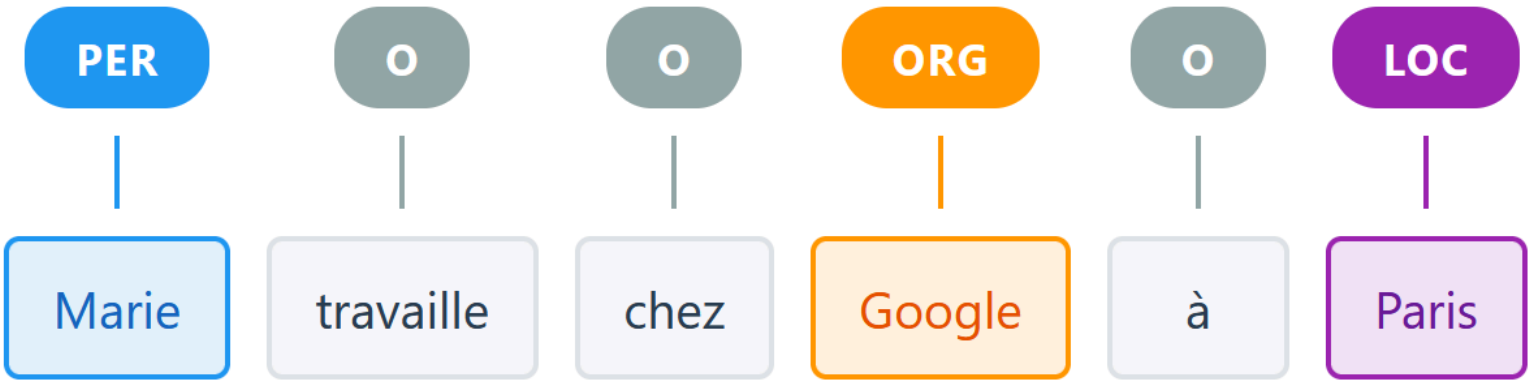
Quelques exemples d'options personnalisées avec *Yake* (voir la **documentation officielle**) :

Python

```python
custom_kw_extractor = yake.KeywordExtractor(
    lan="en",                # langue
    n=3,                     # ngram
    windowsSize=1,           # fenêtre de contexte
    top=10,                  # nb mots-clés à extraire
)
```

Annotation automatique : reconnaissance d'entités nommées

La reconnaissance d'entités nommées (NER) consiste à identifier et classifier des entités nommées (noms de personnes, d'organisations, de lieux, dates, etc.) dans un texte.



Annotation automatique : reconnaissance d'entités nommées

La reconnaissance d'entités nommées (NER) consiste à identifier et classifier des entités nommées (noms de personnes, d'organisations, de lieux, dates, etc.) dans un texte.

Diverses options disponibles :

- spaCy (notebook disponible sur Moodle) ;
- Flair (modèle disponible sur **HuggingFace**) ;
- DistilCamemBERT-NER (modèle disponible sur **HuggingFace** et basé sur CamemBERT).

Annotation automatique : reconnaissance d'entités nommées

Utilisation :

```python
from transformers import pipeline
ner = pipeline(
    task='ner',
    model="cmarkea/distilcamembert-base-ner",
    tokenizer="cmarkea/distilcamembert-base-ner",
    aggregation_strategy="simple" # entités multi-tokens regroupées
)
```

Annotation automatique : reconnaissance d'entités nommées

Exemple de sortie :

```python
[{'entity_group': 'ORG',
  'score': 0.9974479,
  'word': 'Crédit Mutuel Arkéa',
  'start': 3,
  'end': 22},
```

Bibliographie

Boudin, F. (décembre 2016). Pke: An Open Source Python-Based Keyphrase Extraction Toolkit. In H. Watanabe (éd.), *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*.

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., et Jatowt, A. (2018). A Text Feature Based Automatic Keyword Extraction Method for Single Documents. In G. Pasi, B. Piwowarski, L. Azzopardi, et A. Hanbury (Éds.), *Advances in Information Retrieval: Advances in Information Retrieval*. **10.1007/978-3-319-76941-7_63**

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

Manning, C. D., Raghavan, P., et Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Mitra, B. (2018). *An Introduction to Neural Information Retrieval* (Numéro v.41). Now Publishers.

Zhai, C., et Massung, S. (juin 2016). *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. Association for Computing Machinery and Morgan & Claypool. **10.1145/2915031**

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