

▼ Importation

```
# Importation des bibliothèques requises
import requests
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import nltk
from collections import defaultdict
import math
import re

# Téléchargement des ressources NLTK nécessaires
nltk.download('punkt')
nltk.download('stopwords')
```

```
🔄 [nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
# Initialisation du stemmer (racineur) et des mots vides (stopwords)
stemmer = PorterStemmer()
stop_words = set(stopwords.words("english"))
```

▼ Fonction pour récupérer le code HTML d'une URL

```
# Fonction pour récupérer le code HTML d'une URL
def get_html_source(url):
    try:
        response = requests.get(url)
        response.raise_for_status() # Vérifie les erreurs HTTP
        return response.text
    except requests.exceptions.RequestException as e:
        print(f"Erreur lors de la récupération de l'URL : {e}")
        return None
```

▼ Extraction de texte de HTML

```
# Fonction pour extraire le texte brut du code HTML en supprimant les balises
def extract_text_from_html(html_source):
    """Extrait le texte brut du HTML en supprimant les balises."""
    text = re.sub(r'<[^>]+>', '', html_source) # Supprime toutes les balises HTML
    text = re.sub(r'\s+', ' ', text) # Remplace les espaces multiples par un espace unique
    return text.strip()
```

▼ Explication du code pour extraire du texte brut à partir d'un code HTML

Ce code définit une fonction `extract_text_from_html` qui prend en paramètre le contenu source d'une page HTML (`html_source`) et renvoie le texte brut en retirant tous les éléments HTML (balises et espaces inutiles). Voici un aperçu détaillé des étapes du code :

Étapes

1. **Suppression des balises HTML :**
- La première ligne de la fonction utilise l'expression régulière `re.sub(r'<[^>]+>', '', html_source)` pour supprimer toutes les balises HTML du contenu source.
 - L'expression régulière `r'<[^>]+>'` correspond à toutes les chaînes de caractères encadrées par des chevrons (`<...>`), qui sont les balises HTML.
 - En remplaçant ces correspondances par une chaîne vide (`''`), on retire toutes les balises HTML, ne conservant que le texte visible de la page.
2. **Remplacement des espaces multiples :**
- La seconde ligne utilise l'expression régulière `re.sub(r'\s+', ' ', text)` pour remplacer les espaces multiples, y compris les sauts de ligne, par un seul espace.
 - L'expression `\s+` correspond à une ou plusieurs occurrences d'espaces blancs, et en les remplaçant par un seul espace, on assure que le texte final est propre, sans espaces superflus ou sauts de ligne inutiles.
3. **Retour du texte brut nettoyé :**
- La fonction utilise `text.strip()` pour supprimer les espaces en début et en fin de chaîne.
 - Enfin, elle retourne le texte brut, sans balises HTML et avec des espaces bien formatés.

Exemple d'utilisation

Si l'on passe un HTML de la forme `<p>bonjour Monsieur</p>`, la fonction retournera simplement `bonjour Monsieur`, ayant retiré les balises HTML `<p>` et `` et nettoyé le texte.

▼ Traitement de texte (nettoyage, tokenisation, racinisation)

```
def process_text(text):
```

```
tokens = word_tokenize(text) # Tokenise le texte
processed_tokens = []

for word in tokens:
    word = word.lower() # Convertit en minuscule
    if word.isalpha() and word not in stop_words: # Garde uniquement les mots alphabetiques non stopwords
        stemmed_word = stemmer.stem(word) # Racinise le mot
        processed_tokens.append(stemmed_word)

return processed_tokens
```

Construction de Fichier Inversé

```
# Fonction pour construire l'index inversé
def build_inverted_index(urls):
    inverted_index = {}
    total_documents = len(urls)
# Nombre total de documents pour le calcul du poids (poids)

    for doc_id, url in enumerate(urls):
        print(f"Indexation de l'ID d'URL {doc_id}: {url}")
        html_source = get_html_source(url)

        if html_source:
            text = extract_text_from_html(html_source)
            tokens = process_text(text)

            for word in tokens:
                if word not in inverted_index:
                    inverted_index[word] = {"df": 0, "List_Posting": {}, "poids": {}}

                # Mise à jour de la fréquence de document (df)
                if doc_id not in inverted_index[word]["List_Posting"]:
                    inverted_index[word]["df"] += 1
                    inverted_index[word]["List_Posting"][doc_id] = 1
                else:
                    # Mise à jour de la fréquence de terme (tf) pour ce mot dans ce document
                    inverted_index[word]["List_Posting"][doc_id] += 1

# Calcul du poids pour chaque terme dans chaque document
for term, data in inverted_index.items():
    df = data["df"]
    for doc_id, tf in data["List_Posting"].items():
        poids = tf * math.log(total_documents / df)
        inverted_index[term]["poids"][doc_id] = poids


return inverted_index
```

Liste des URLs

```
# Liste des URLs à indexer
urls = [
    "https://www.musicca.com/note-finder",
    "https://www.musicca.com/interval-finder",
    "https://www.musicca.com/chord-finder",
    "https://www.musicca.com/scale-finder",
    "https://www.musicca.com/piano",
    "https://www.musicca.com/guitar",
    "https://www.musicca.com/bass-guitar",
    "https://www.musicca.com/drums", "https://www.musicca.com/exercises/notes",
    "https://www.musicca.com/exercises/rhythms",
    "https://www.musicca.com/exercises/intervals",
    "https://www.musicca.com/exercises/chords",
    "https://www.musicca.com/exercises/scales",
    "https://www.musicca.com/exercises/key-signatures",
    "https://www.musicca.com/exercises/instruments",
    "https://www.musicca.com/exercises/genres"
]
```

Construction de Fichier Inversé

```
inverted_index = build_inverted_index(urls)
```

 Indexation de l'ID d'URL 0: <https://www.musicca.com/note-finder>
Indexation de l'ID d'URL 1: <https://www.musicca.com/interval-finder>
Indexation de l'ID d'URL 2: <https://www.musicca.com/chord-finder>
Indexation de l'ID d'URL 3: <https://www.musicca.com/scale-finder>
Indexation de l'ID d'URL 4: <https://www.musicca.com/piano>
Indexation de l'ID d'URL 5: <https://www.musicca.com/guitar>
Indexation de l'ID d'URL 6: <https://www.musicca.com/bass-guitar>
Indexation de l'ID d'URL 7: <https://www.musicca.com/drums>
Indexation de l'ID d'URL 8: <https://www.musicca.com/exercises/notes>
Indexation de l'ID d'URL 9: <https://www.musicca.com/exercises/rhythms>
Indexation de l'ID d'URL 10: <https://www.musicca.com/exercises/intervals>
Indexation de l'ID d'URL 11: <https://www.musicca.com/exercises/chords>
Indexation de l'ID d'URL 12: <https://www.musicca.com/exercises/scales>
Indexation de l'ID d'URL 13: <https://www.musicca.com/exercises/key-signatures>
Indexation de l'ID d'URL 14: <https://www.musicca.com/exercises/instruments>
Indexation de l'ID d'URL 15: <https://www.musicca.com/exercises/genres>

```
inverted_index
```

```
'second': {'df': 1,
'List_Posting': {10: 1},
'poids': {10: 2.772588722239781}},
'third': {'df': 1, 'List_Posting': {10: 2}, 'poids': {10: 5.545177444479562}},
'fourth': {'df': 1,
'List_Posting': {10: 1},
'poids': {10: 2.772588722239781}},
'fifth': {'df': 1, 'List_Posting': {10: 3}, 'poids': {10: 8.317766166719343}},
'unison': {'df': 1,
'List_Posting': {10: 2},
'poids': {10: 5.545177444479562}},
'sixth': {'df': 1, 'List_Posting': {10: 1}, 'poids': {10: 2.772588722239781}},
'octav': {'df': 1,
'List_Posting': {10: 4},
'poids': {10: 11.090354888959125}},
'extend': {'df': 3,
'List_Posting': {10: 1, 11: 1, 12: 1},
'poids': {10: 1.6739764335716716,
11: 1.6739764335716716,
12: 1.6739764335716716}},
'ear': {'df': 3,
'List_Posting': {10: 1, 11: 1, 12: 1},
'poids': {10: 1.6739764335716716,
11: 1.6739764335716716,
12: 1.6739764335716716}},
'dim': {'df': 1, 'List_Posting': {11: 1}, 'poids': {11: 2.772588722239781}},
'aug': {'df': 1, 'List_Posting': {11: 2}, 'poids': {11: 5.545177444479562}},
'triad': {'df': 1, 'List_Posting': {11: 1}, 'poids': {11: 2.772588722239781}},
'tetrad': {'df': 1,
'List_Posting': {11: 1},
'poids': {11: 2.772588722239781}},
'pentad': {'df': 1,
'List_Posting': {11: 1},
'poids': {11: 2.772588722239781}},
'natur': {'df': 1, 'List_Posting': {12: 1}, 'poids': {12: 2.772588722239781}},
'pentaton': {'df': 1,
'List_Posting': {12: 1},
'poids': {12: 2.772588722239781}},
'blue': {'df': 1, 'List_Posting': {12: 2}, 'poids': {12: 5.545177444479562}},
'mode': {'df': 1, 'List_Posting': {12: 2}, 'poids': {12: 5.545177444479562}},
'three': {'df': 1, 'List_Posting': {13: 3}, 'poids': {13: 8.317766166719343}},
'common': {'df': 1,
'List_Posting': {14: 2},
'poids': {14: 5.545177444479562}},
'popular': {'df': 2,
'List_Posting': {14: 1, 15: 1},
'poids': {14: 2.0794415416798357, 15: 2.0794415416798357}},
'classic': {'df': 2,
'List_Posting': {14: 1, 15: 1},
'poids': {14: 2.0794415416798357, 15: 2.0794415416798357}},
'percuss': {'df': 1,
'List_Posting': {14: 1},
'poids': {14: 2.772588722239781}},
'wind': {'df': 1, 'List_Posting': {14: 1}, 'poids': {14: 2.772588722239781}},
'jazz': {'df': 1, 'List_Posting': {15: 1}, 'poids': {15: 2.772588722239781}},
'period': {'df': 1,
'List_Posting': {15: 1},
'poids': {15: 2.772588722239781}}}
```

len(inverted_index)

318

Le filtrage avec le seuil de Poids

```
# Fonction pour construire l'index inversé avec filtrage par seuil de poids
def build_inverted_index_filtre(urls, seuil_poids=2):
    inverted_index = {}
    total_documents = len(urls)
    # Nombre total de documents pour le calcul du poids

    # Parcours des URLs pour construire l'index inversé
    for doc_id, url in enumerate(urls):
        print(f"Indexation de l'ID d'URL {doc_id}: {url}")
        html_source = get_html_source(url)

        if html_source:
            text = extract_text_from_html(html_source)
            tokens = process_text(text)

            # Comptabilisation des fréquences de termes
            for word in tokens:
                if word not in inverted_index:
                    inverted_index[word] = {"df": 0, "List_Posting": {}, "poids": {}}

                # Mise à jour de la fréquence de document (df) et fréquence de terme (tf)
                if doc_id not in inverted_index[word]["List_Posting"]:
                    inverted_index[word]["df"] += 1
                    inverted_index[word]["List_Posting"][doc_id] = 1
                else:
                    inverted_index[word]["List_Posting"][doc_id] += 1

    # Calcul du poids pour chaque terme dans chaque document
    for term, data in list(inverted_index.items()):
        df = data["df"]
        for doc_id, tf in list(data["List_Posting"].items()):
            poids = tf * math.log(total_documents / df)
            # Appliquer le poids seulement si supérieur ou égal au seuil
            if poids >= seuil_poids:
                inverted_index[term]["poids"][doc_id] = poids
            else:
                # Supprimer le terme du document si le poids est inférieur au seuil
                del inverted_index[term]["List_Posting"][doc_id]

    # Supprimer le terme de l'index si aucun document ne le conserve
```

```
if not inverted_index[term]["List_Posting"]:\n    del inverted_index[term]\n\nreturn inverted_index\n\n# Construction de l'index inversé avec un seuil de poids de 2\ninverted_index_filtre = build_inverted_index_filtre(urls, seuil_poids=2)
```

```
➤ Indexation de l'ID d'URL 0: https://www.musicca.com/note-finder\nIndexation de l'ID d'URL 1: https://www.musicca.com/interval-finder\nIndexation de l'ID d'URL 2: https://www.musicca.com/chord-finder\nIndexation de l'ID d'URL 3: https://www.musicca.com/scale-finder\nIndexation de l'ID d'URL 4: https://www.musicca.com/piano\nIndexation de l'ID d'URL 5: https://www.musicca.com/guitar\nIndexation de l'ID d'URL 6: https://www.musicca.com/bass-guitar\nIndexation de l'ID d'URL 7: https://www.musicca.com/drums\nIndexation de l'ID d'URL 8: https://www.musicca.com/exercises/notes\nIndexation de l'ID d'URL 9: https://www.musicca.com/exercises/rhythms\nIndexation de l'ID d'URL 10: https://www.musicca.com/exercises/intervals\nIndexation de l'ID d'URL 11: https://www.musicca.com/exercises/chords\nIndexation de l'ID d'URL 12: https://www.musicca.com/exercises/scales\nIndexation de l'ID d'URL 13: https://www.musicca.com/exercises/key-signatures\nIndexation de l'ID d'URL 14: https://www.musicca.com/exercises/instruments\nIndexation de l'ID d'URL 15: https://www.musicca.com/exercises/genres
```

```
inverted_index_filtre\n➤\n    12: 2.6228541460898747,\n    14: 3.746934494414107,\n    15: 2.248160696648464}},\n'f': {'df': 2,\n      'List_Posting': {4: 1, 7: 1},\n      'poids': {4: 2.0794415416798357, 7: 2.0794415416798357}},\n'minor': {'df': 3,\n          'List_Posting': {11: 2, 12: 3},\n          'poids': {11: 3.347952867143343, 12: 5.021929300715015}},\n'seventh': {'df': 3,\n            'List_Posting': {10: 2},\n            'poids': {10: 3.347952867143343}},\n'major': {'df': 5, 'List_Posting': {5: 2}, 'poids': {5: 2.3263016196113617}},\n'help': {'df': 2,\n         'List_Posting': {4: 1, 7: 1},\n         'poids': {4: 2.0794415416798357, 7: 2.0794415416798357}},\n'becom': {'df': 2,\n          'List_Posting': {4: 1, 7: 1},\n          'poids': {4: 2.0794415416798357, 7: 2.0794415416798357}},\n'better': {'df': 2,\n           'List_Posting': {4: 1, 7: 1},\n           'poids': {4: 2.0794415416798357, 7: 2.0794415416798357}},\n'lowest': {'df': 1, 'List_Posting': {5: 1}, 'poids': {5: 2.772588722239781}},\n'highlight': {'df': 2,\n              'List_Posting': {5: 1, 6: 1},\n              'poids': {5: 2.0794415416798357, 6: 2.0794415416798357}},\n'choos': {'df': 1, 'List_Posting': {5: 1}, 'poids': {5: 2.772588722239781}},\n'differ': {'df': 1, 'List_Posting': {5: 1}, 'poids': {5: 2.772588722239781}},\n'sound': {'df': 2,\n          'List_Posting': {5: 4, 6: 3},\n          'poids': {5: 8.317766166719343, 6: 6.238324625039507}},\n'fretboard': {'df': 2,\n              'List_Posting': {5: 1, 6: 2},\n              'poids': {5: 2.0794415416798357, 6: 4.1588830833596715}},\n'one': {'df': 2,\n        'List_Posting': {5: 2, 6: 1},\n        'poids': {5: 4.1588830833596715, 6: 2.0794415416798357}},\n'alt': {'df': 2,\n        'List_Posting': {5: 1, 6: 1},\n        'poids': {5: 2.0794415416798357, 6: 2.0794415416798357}},\n'c': {'df': 2,\n      'List_Posting': {5: 1, 7: 1},\n      'poids': {5: 2.0794415416798357, 7: 2.0794415416798357}},\n'tuner': {'df': 2,\n          'List_Posting': {5: 4, 6: 4},\n          'poids': {5: 8.317766166719343, 6: 8.317766166719343}},\n'tune': {'df': 2,\n         'List_Posting': {5: 4, 6: 4},\n         'poids': {5: 8.317766166719343, 6: 8.317766166719343}},\n'devic': {'df': 2,\n          'List_Posting': {5: 1, 6: 1},\n          'poids': {5: 2.0794415416798357, 6: 2.0794415416798357}},\n'microphon': {'df': 2,\n              'List_Posting': {5: 1, 6: 1},\n              'poids': {5: 2.0794415416798357, 6: 2.0794415416798357}},\n'activ': {'df': 2,\n          'List_Posting': {5: 1, 6: 1},\n          'poids': {5: 2.0794415416798357, 6: 2.0794415416798357}},\n'manual': {'df': 2,
```

```
len(inverted_index_filtre)\n➤ 166
```

On remarque une diminution de taille lorsque on fait le filtrage

+ Code

+ Texte

```
print("Taille de inverted_index:",len(inverted_index))\nprint("Taille de inverted_index_filtre:",len(inverted_index_filtre))
```

```
➤ Taille de inverted_index: 318\n  Taille de inverted_index_filtre: 166
```

