TD 1 — Initiation pratique au TAL : analyse stylistique et linguistique avec Python

Niveau : Master 2 — Linguistique
Outil : Python avec Thonny
Durée indicative : 1 h 30
Thème : Comparaison linguistique de textes littéraires.

1. Objectifs du TD

À la fin de ce TD, l'étudiant sera capable de :

- utiliser Python pour analyser des textes littéraires ;
- identifier automatiquement les catégories grammaticales ;
- calculer des indicateurs linguistiques simples ;
- comparer les caractéristiques de deux textes ;
- adapter un programme Python existant ;
- utiliser un LLM pour générer du code, puis vérifier son fonctionnement et ses résultats.

Exercice 1 — Démonstration : analyser et comparer deux textes

1.1. Problématique

En analyse stylistique, on peut chercher à identifier certaines caractéristiques linguistiques d'un texte :

- la longueur des phrases ;
- la fréquence des noms, des verbes et des adjectifs ;
- l'utilisation des pronoms ;
- la diversité du vocabulaire.

Ces caractéristiques peuvent ensuite être comparées entre deux textes.

Prenons deux courts textes fictifs, créés uniquement pour cette démonstration.

Texte A :

> Le soir tombe sur la ville. Les rues deviennent silencieuses. Un homme marche lentement. Il regarde les fenêtres éclairées et pense à son enfance.

Texte B :

> Lorsque la nuit recouvre progressivement la ville, l'homme qui avance dans les rues désertes observe les fenêtres illuminées, tandis que ses souvenirs d'enfance remontent lentement à sa conscience.

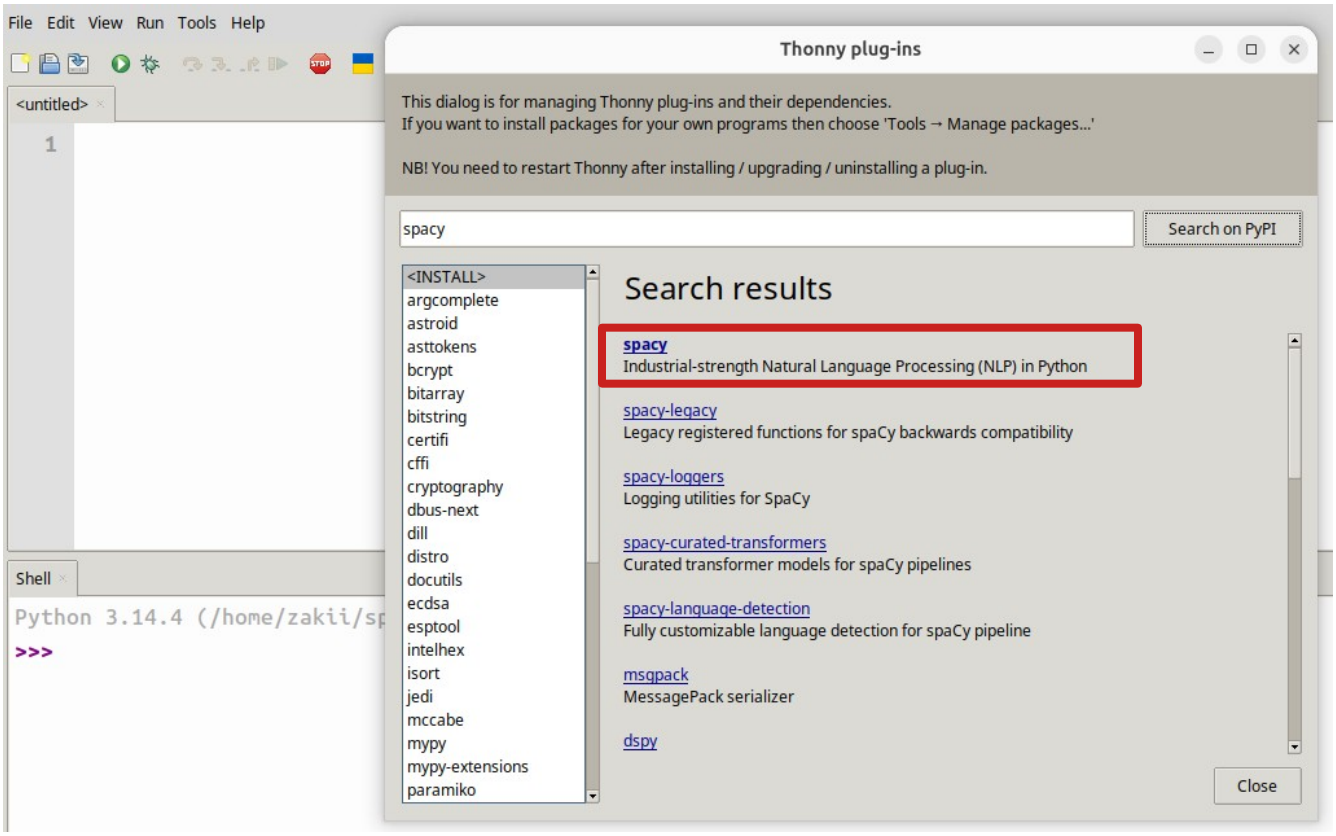
Notre objectif est de déterminer si ces deux textes présentent des différences mesurables.

1.2. Première étape : préparer l'environnement Python

Pour analyser automatiquement la grammaire française, nous allons utiliser la bibliothèque **spaCy**.

Si elle est déjà installée dans l'environnement Python utilisé par Thonny, cette étape n'est pas nécessaire.

Installer la bibliothèque Spacy dans votre environnement de développement (Dans note cas le logiciel Thonny), Cliquer sur “Tools”, ensuite “Manage Plug-ins” .. dans l’interface qui apparait, chercher et installer la bibliothèque Spacy :



1.3. Deuxième étape : écrire le programme

Dans Thonny, créer un nouveau fichier et saisir le code suivant.

```
import spacy
from collections import Counter
# Charger le modèle linguistique français
nlp = spacy.load("fr_core_news_sm")

# Texte A
texte_a = """
Le soir tombe sur la ville.
Les rues deviennent silencieuses.
Un homme marche lentement.
Il regarde les fenêtres éclairées et pense à son enfance.
"""

# Texte B
texte_b = """
Lorsque la nuit recouvre progressivement la ville,
l'homme qui avance dans les rues désertes observe
les fenêtres illuminées, tandis que ses souvenirs
d'enfance remontent lentement à sa conscience.
"""

# Analyser les deux textes
doc_a = nlp(texte_a)
doc_b = nlp(texte_b)



# Fonction d'analyse linguistique
def analyser_texte(doc):

```

```
    # Compter les phrases
    phrases = list(doc.sents)
    nombre_phrases = len(phrases)

    # Sélectionner les mots
    mots = [token for token in doc if token.is_alpha]
    nombre_mots = len(mots)

    # Compter les catégories grammaticales (étiquetage morphosyntaxique.)
    categories = Counter(token.pos_ for token in mots)

    # Calculer la longueur moyenne des phrases
    if nombre_phrases > 0:
        longueur_moyenne = nombre_mots / nombre_phrases
    else:
        longueur_moyenne = 0

    # Calculer la diversité lexicale (on applique d’abord la lemmatisation)
    lemmes = [
        token.lemma_.lower()
        for token in mots
    ]

    if nombre_mots > 0:
        diversite = len(set(lemmes)) / nombre_mots
    else:
        diversite = 0

    return {
        "phrases": nombre_phrases,
        "mots": nombre_mots,
        "longueur_moyenne": longueur_moyenne,
```

```
        "adjectifs": categories["ADJ"],
        "noms": categories["NOUN"],
        "verbes": categories["VERB"],
        "pronoms": categories["PRON"],
        "diversite_lexicale": diversite
    }


# Analyser les textes
resultat_a = analyser_texte(doc_a)
resultat_b = analyser_texte(doc_b)


# Afficher les résultats
print("===== TEXTE A =====")

for indicateur, valeur in resultat_a.items():
    print(indicateur, ":", round(valeur, 2)
          if isinstance(valeur, float) else valeur)

print("\n===== TEXTE B =====")

for indicateur, valeur in resultat_b.items():
    print(indicateur, ":", round(valeur, 2)
          if isinstance(valeur, float) else valeur)
```

1.4. Troisième étape : exécuter le programme

Le programme affiche, pour chaque texte :

Indicateur	Signification
`phrases`	Nombre de phrases détectées
`mots`	Nombre de tokens reconnus comme mots alphabétiques
`longueur_moyenne`	Nombre moyen de mots par phrase

`adjectifs`	Nombre d'adjectifs identifiés
`noms`	Nombre de noms communs identifiés
`verbes`	Nombre de verbes identifiés
`pronoms`	Nombre de pronoms identifiés
`diversite_lexicale`	Proportion de lemmes distincts parmi les mots retenus

Les nombres exacts sont calculés par le programme et apparaissent dans la console de Thonny.

1.5. Quatrième étape : interpréter les résultats

1. Quel texte possède le plus grand nombre de phrases ?
2. Quel texte possède la longueur moyenne de phrase la plus élevée ?
3. Les deux textes utilisent-ils les mêmes proportions de noms, de verbes et d'adjectifs ?
4. Que mesure la diversité lexicale ?
5. Peut-on affirmer que le texte possédant les phrases les plus longues est nécessairement plus littéraire ?
6. Peut-on identifier le style d'un auteur à partir de ces seuls indicateurs ?

Conclusion :

Python permet de transformer des observations linguistiques en indicateurs mesurables. Ces indicateurs peuvent aider à comparer des textes, mais ils ne suffisent pas, à eux seuls, à expliquer leur sens ou leur valeur littéraire.

De plus, les résultats dépendent notamment de la qualité de l'analyse automatique effectuée par spaCy. Les catégories grammaticales identifiées doivent être vérifiées lorsque la précision est importante.

Exercice 2 — À vous de jouer : comparer deux textes littéraires

2.1. Problématique

Vous êtes chargé d'effectuer une première comparaison linguistique entre deux textes littéraires.

Vous devez utiliser Python pour extraire des caractéristiques linguistiques, puis interpréter les résultats.

Vous pouvez choisir deux courts extraits de Camus et de Sartre.

L'objectif est de rechercher des différences observables, sans supposer à l'avance qu'un auteur utilise nécessairement plus de noms, d'adjectifs ou de phrases longues que l'autre.

2.2. Travail demandé

À partir du programme de l'exercice 1, réalisez les tâches suivantes :

Tâche 1 — Modifier les textes

Remplacez les textes A et B par les deux extraits que vous souhaitez analyser.

Tâche 2 — Comparer les résultats

Exécutez le programme et relevez les indicateurs suivants :

- nombre de phrases ;
- longueur moyenne des phrases ;
- nombre d'adjectifs ;
- nombre de noms ;
- nombre de verbes ;
- diversité lexicale.

Tâche 3 — Ajouter un nouvel indicateur

Modifiez le programme pour calculer le nombre de **noms et de verbes réunis** dans chaque texte.

Vous pouvez vous inspirer de cette instruction :

```
nombre_noms_verbes = categories["NOUN"] + categories["VERB"]
```

Ajoutez cet indicateur au résultat retourné par la fonction `analyser_texte()`.

Tâche 4 — Interpréter

Rédigez une courte comparaison de 5 à 8 lignes présentant :

- les différences linguistiques observées ;
- les indicateurs qui vous semblent les plus intéressants ;
- une interprétation possible de ces différences ;
- au moins une limite de votre analyse automatique.

2.3. Aide possible avec un LLM

Si vous rencontrez des difficultés pour modifier le programme, vous pouvez demander de l'aide à un LLM.

Exemple de prompt :

> Je suis étudiant en Master 2 Linguistique et j'utilise Python avec Thonny et spaCy pour comparer deux textes littéraires.

> Voici mon programme Python : [coller le programme de l'exercice 1].
>
> Je souhaite ajouter un indicateur qui calcule le nombre de noms et de verbes réunis dans chaque texte.
>
> Modifie mon programme en conservant sa structure et explique précisément les lignes ajoutées. N'ajoute pas de fonctionnalités inutiles.

Attention : le code généré par un LLM doit être testé dans Thonny. Une explication convaincante ne garantit pas que le programme fonctionne correctement.

2.4. Vérification du programme

Après avoir modifié ou généré le code, vérifiez les points suivants :

1. Le programme s'exécute-t-il sans erreur ?
2. Les deux textes sont-ils bien analysés ?
3. Le nombre de noms et de verbes correspond-il à la somme des deux catégories affichées ?
4. Les catégories grammaticales sont-elles toutes correctement identifiées ?
5. Les résultats permettent-ils réellement de soutenir votre interprétation ?

2.5. Travail à remettre

Chaque étudiant remet une courte fiche comprenant :

- les deux textes analysés ;
- le programme Python modifié ;

Bilan du TD

Ce premier TD montre comment un programme TAL peut aider à analyser des textes littéraires à partir d'indicateurs linguistiques.

Il met également en évidence trois principes importants :

1. **Automatiser une analyse ne signifie pas automatiser toute l'interprétation littéraire.**
2. **Un résultat produit par un programme doit être vérifié avant d'être utilisé dans une recherche.**
3. **Un LLM peut aider à programmer, mais le chercheur doit tester le code, examiner les résultats et justifier sa méthode.**

Dans les prochains TD, ces principes pourront être appliqués à des tâches plus avancées : extraction d'entités nommées, analyse du vocabulaire, classification de textes ou comparaison de corpus littéraires.