Programmation de Modèles Linguistiques (I)

(L5SOPROG L3 Sciences du Langage)

2024-2025

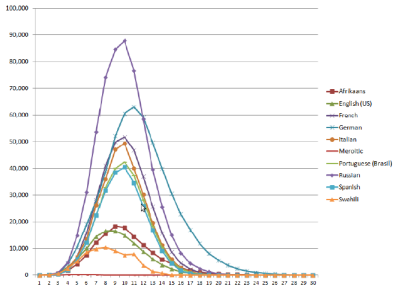
TD2 Analyse en caractères II

Caroline Koudoro-Parfait, Sorbonne Université.
Crédit : Gaël Lejeune, Sorbonne Université

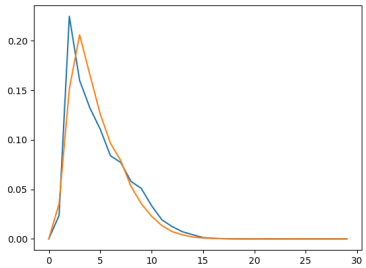
Objectifs

- Comprendre la factorisation
- Améliorer un code et un rendu
- Procéder à une analyse de données

Exercice 1 : Rappel de l'attendu



(a) La figure recherchée



(b) Là où nous nous sommes arrêtés

La première ligne qu'il faut améliorer c'est la sélection des fichiers, en effet si l'on doit traiter 1.000 fichiers on ne va pas écrire leurs noms à la main. Ici on va faire deux choses:

- regrouper nos textes dans un dossier "data"
- utiliser une libraire qui va nous permettre de lister les textes de ce dossier :

```
import glob #la bonne librairie
print(glob.glob("./*")) #le contenu dossier courant
print(glob.glob("data/*")) #le contenu du dossier "data"
```

Le code ci-dessous suppose donc d'avoir des fichiers dans un dossier data situé dans le même dossier que votre code. **Si rien ne s'affiche c'est que Python ne trouve rien dans votre dossier data** (vraisemblablement il est vide il n'est pas situé au même endroit que votre code.

```
import matplotlib.pyplot as pyplot
for chemin in glob.glob("data/*"):
  print(chemin)
  chaine = lire_fichier(chemin)
  liste_mots = decouper_en_mots(chaine)
```

```
  dic_longueurs = get_effectifs(liste_mots)
  liste_effectifs = vecteur_longueurs(dic_longueurs)
  pyplot.plot(liste_effectifs)
pyplot.savefig("frequences.png")
pyplot.show()
```

Maintenant nous allons ajouter des étiquettes aux axes (*labels*), un titre (*title*), et une légende (*legend*). Ce qui va nous donner la Figure 2.

```
...
for chemin in glob.glob("../data/*"):
  print(chemin)
  nom_pour_legende = chemin
 ....
  #Ici ne pas oublier d'ajouter les étapes de traitement
  pyplot.plot(liste_effectifs, label = nom_pour_legende)
pyplot.legend()
pyplot.title("Une magnifique Courbe")
pyplot.xlabel("Longueur des Mots")
pyplot.ylabel("Fréquence")
pyplot.savefig("frequences2.png")
pyplot.show()
```

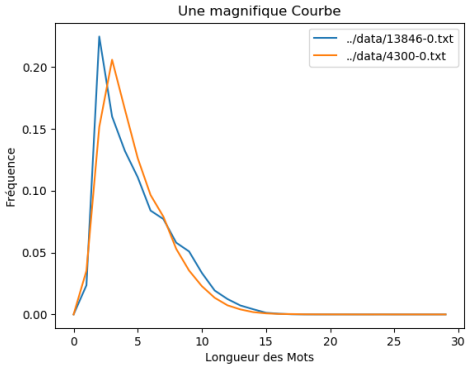


Figure 2: Avec la légende

Pour rendre la légende de la figure plus explicite, on va modifier le nom des fichiers (à la main) pour faire apparaître la langue sous la forme d'un code à deux lettres [1]. Nos fichiers se nomment donc désormais :

- `en_4300-0.txt`
- `fr_13846-0.txt`

Puis nous allons formater le nom du fichier pour que seul le nom de la langue apparaisse dans la légende au moyen d'une expression régulière (attention à la librairie):

Si mon dossier s'appelle bein *data*, alors le nom de la langue se situe entre le caractère 5 et le caractère 7. Le tableau 1 ci-dessous montre la correspondance entre les positions et les caractères dans le chemin d'un de mes fichiers.

[1] Norme ISO : `https://fr.wikipedia.org/wiki/Liste_des_codes_ISO_639-1`

Position	0	1	2	3	4	**5**	**6**	7	8	9	10	11	12	13	14	15	16	17
Caractère	d	a	t	a	/	**e**	**n**	_	4	3	0	0	-	0	.	t	x	t

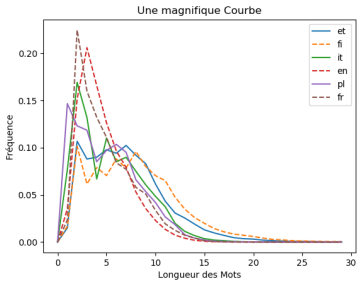
Table 1: Repérage du code langue dans le chemin en fonction de la position des lettres dans la chaîne

```
nom_pour_legende = chemin[5:7]
```

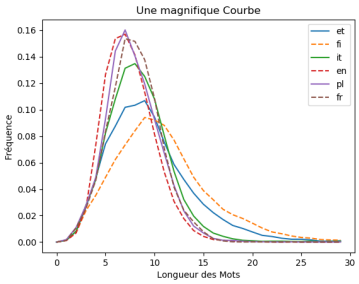
Pour avoir une idée du rendu, regardez la légende de la Figure 3a.

Exercice 2 : Devoir à faire et à déposer en TD

Nous allons ajouter de nouveaux textes, récupérés sur `gutenberg.org` par exemple, pour avoir une figure avec plusieurs langues. Attention, il faut avoir des donnéess suffisamment grands pour être significatifs, au moins 100.000 mots soit 500 Ko. Au besoin, concaténez (fusionnez) plusieurs textes pour une même langue.



(a) Nouvelle courbe avec plus de langues



(b) Changement de la sortie de `decouper_mots`

Efforcez vous d'avoir des données pour plusieurs grandes familles de langues ([2]) et plusieurs types de langues[3]). Votre corpus devra contenir au moins 10 langues avec au moins 3 familles et 3 types différents. Vous produirez une autre figure avec les mêmes données mais où vous aurez simplement changé la sortie de la fonction `decouper_mots` de la façon suivante :

```
return set(liste_mots)
```

Ce qui devrait vous donner quelque chose qui ressemble à la Figure 3b.

Devoir

Vous produirez un document d'une page (à l'aide du traitement de textes de votre choix) environ incluant :

- des statistiques sur votre corpus (nombre de caractères et de mots pour chaque langue)
- les figures que vous avez généré
- des commentaires sur ces figures et leurs différences
- quelques phrases de conclusion (qu'est-ce qui était attendu, qu'est-ce qui est inattendu...)

Vous déposerez sur Moodle une archive zip nommée NUMETU.zip (où NUMETU est votre numéro d'étudiant) et contenant :

- Votre code exporté au format Python `.py` (et pas ipynb)
- Des commentaires (précédés de "#") pour chaque fonction utilisée et chaque cellule
- le PDF du document que vous avez produit

Date limite : cf Moodle **La suite page suivante...**

[2] `https://fr.wikipedia.org/wiki/Langues_par_famille`
[3] `https://fr.wikipedia.org/wiki/Typologie_morphologique_des_langues`

Exercice bonus : une fois que le travail est terminé et déposé, essayez de trouver un moyen d'améliorer la fonction `decouper_mots` en tenant compte de la langue. Il s'agit de chercher un tokenizer (découpeur en mots) adapté pour chaque langue. Considérez l'exemple suivant (il faut au préalable installer la librairie MosesTokenizer[4] vir par exemple la commande `pip install mosestokenizer` dans le terminal. Ce qui nécessite d'installer `pip` si vous ne l'avez pas déjà[5].

Créez la variable suivante: phrase = "L'élision est l'effacement d'une voyelle en fin de mot devant la voyelle commençant le mot suivant."

Et testez ce code :

```
mots_base = phrase.split()
print(mots_base)#tokenisation de base
from mosestokenizer import *
tokenize = MosesTokenizer(lang='fr')
mots = tokenize(phrase)
print(mots)#tokenisation plus correcte

print("Mots obtenus avec split mais avec Moses:")
print(set(mots_base).difference(set(mots)))

print("L'inverse:")
print(set(mots).difference(set(mots_base))

print("Les mots en commun (l'intersection)")
print(set(mots).intersection(set(mots_base))
```

Attention, ça ne marchera peut être pas parfaitement pour toutes les langues !

[4] `https://pypi.org/project/mosestokenizer/`

[5] https://pip.pypa.io/en/stable/installing/