TraAM IA et Esprit Critique

Cette activité vous est proposé par le groupe TraAM IA et Esprit Critique constitué sur les années 2024-2026 de la région académique Provence-Alpes-Côte d'Azur (académie de Nice et Marseille).

Ceci est un article de présentation.

Vous trouverez en pièce jointe 6 fichier :

- l'intégrale des pièces de Molière et des Misérables de Victor Hugo. Ces textes sont extrait du projet gutemberg (https://www.gutenberg.org/browse/languages/fr) d'ouvrage en libre accès. Ces textes sont presques complets : pour chacun un extrait en a été supprimé ;
- un fichier "mystère1" provenant de l'auteur originel ;
- un fichier "mystère2" provenant de ChatGPT avec un prompt assez simple "écrire un passage à la manière de ..."

Dans ce carnet, on vous propose de travailler avec les textes de Victor Hugo.

L'objectif est de faire une analyse statistique des différents passages pour savoir s'il est possible de détecter si l'oeuvre provient ou non d'une intelligence artificielle : autrement dit, les élèves, eux, ne savent pas si c'est mystere1 ou mystere2 qui a été généré par l'IA. Cette activité cible des élèves de niveau 1ere en mathématique ou NSI. La structure de liste est en effet utilisée régulièrement (et peut être adaptée à une structure de dictionnaire en NSI) et n'est au programme qu'à partir de la classe de 1ere.

Le code, complet et expliqué, est à adapter à vos élèves suivant leur niveau. Des propositions vous sont faites par moment. L'activité 1cf5-3132490 vous propose un code adapté à des élèves de 1ere NSI et joué en classe.

Entrée[1]:

```
# import des librairies utiles

# la librairie string permet de travailler rapidement sur la ponctuation, on pourrait s'en passer avec une chaine
# caractère du type ",?;.:!'"
import string

# la bibliothèque matplotlib permettra de faire des graphiques. La bibliothèque numpy permet de faire les axes.
import matplotlib.pyplot as plt
import numpy as np

# la bibliothèque statistics permet de faire des calculs de moyenne et écart-type (pstdev).
# ces fonctions peuvent aussi être programmer à la main par les élèves, depuis la classe de 2nde.
from statistics import mean, pstdev
```

Ouvrir un fichier texte

Nous proposons d'ouvrir et de formater le texte présent dans le fichier texte. En mathématique, le code sera donné quasi intégralement, en NSI, il pourra être demandé ou être à compléter.

Entrée[2]:

```
def ouvrir_fichier_texte(nom_du_fichier):
    "Ouvrir un fichier texte et l'avoir en minuscules"
    # on crée une liste vide
    liste = []
    # on ouvre le fichier en "read" = lecture
    with open(nom_du_fichier,'r',encoding='utf-8') as mon_fichier:
        # On enregistre le contenu à la lecture
        contenu = mon_fichier.read()
    # Suppression des ponctuations, que l'on remplace par un espace " ".
    ponctuations = string.punctuation
    for p in ponctuations:
        contenu = contenu.replace(p," ")
    #suppresion des retours à la ligne, que l'on remplace par un espace " ".
    contenu = contenu.replace("\n"," ")
    # gestion des accents
    contenu = contenu.replace("é","e")
    contenu = contenu.replace("è","e")
    contenu = contenu.replace("ê","e")
    contenu = contenu.replace("à","a")
    contenu = contenu.replace("ù","u")
    contenu = contenu.replace("ç","c")
    # le texte est renvoyé en minuscule
    return contenu.lower()
```

Ouverture des fichiers :

Entrée[3]:

```
hugo = ouvrir_fichier_texte("Hugo_miserable.txt")
hugo_mystere1 = ouvrir_fichier_texte("Hugo_mystere1.txt")
hugo_mystere2 = ouvrir_fichier_texte("Hugo_mystere2.txt")
moliere = ouvrir_fichier_texte("Moliere_integrale.txt")
moliere_mystere1 = ouvrir_fichier_texte("Moliere_mystere1.txt")
moliere_mystere2 = ouvrir_fichier_texte("Moliere_mystere2.txt")
```

Travail sur les lettres

Dans cette première partie, nous proposons de travailler sur les lettres utilisées.
Pour cela, on crée la liste des fréquences d'apparitions de chaque lettre :

- on initialise le nombre de lettre de l'alphabet parcouru à 0 ;
- on parcours le texte sur ses lettres : pour chaque lettre de ce texte :
  - si c'est bien une lettre de l'alphabet ;
    - on cherche son numéro dans l'alphabet ;
    - on indique que l'on a trouvé cette lettre une fois de plus ;
    - on indique que l'on a trouvé une lettre de l'alphabet de plus.
- le parcours étant fini, il suffit de calculer la fréquence avec la formule "apparition de la lettre / nombre de lettres"

Entrée[4]:

```
ALPHABET = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z

def frequence_apparition(texte:str) -> list:
    "Donne la liste des frequences des lettres dans le texte"
    reponse = [0 for i in ALPHABET] # création d'une liste de 26 zéros
    nbre_lettres = 0 # initialisation du nombre de lettres
    for lettre in texte:
        if lettre in ALPHABET: # pour la gestion des caractères comme espace ou autre
            numero_lettre = ALPHABET.index(lettre) # place de la lettre dans l'alphabet, une boucle est possible.
            reponse[numero_lettre] = reponse[numero_lettre] + 1
            nbre_lettres = nbre_lettres + 1
    # On a obtenu le nombre de chaque lettre, on peut calculer la fréquence à 10^(-4)
    for i in range(26):
        reponse[i] = round(reponse[i] / nbre_lettres,4)
    # On renvoie la liste
    return reponse
```

Application aux textes :

Entrée[5]:

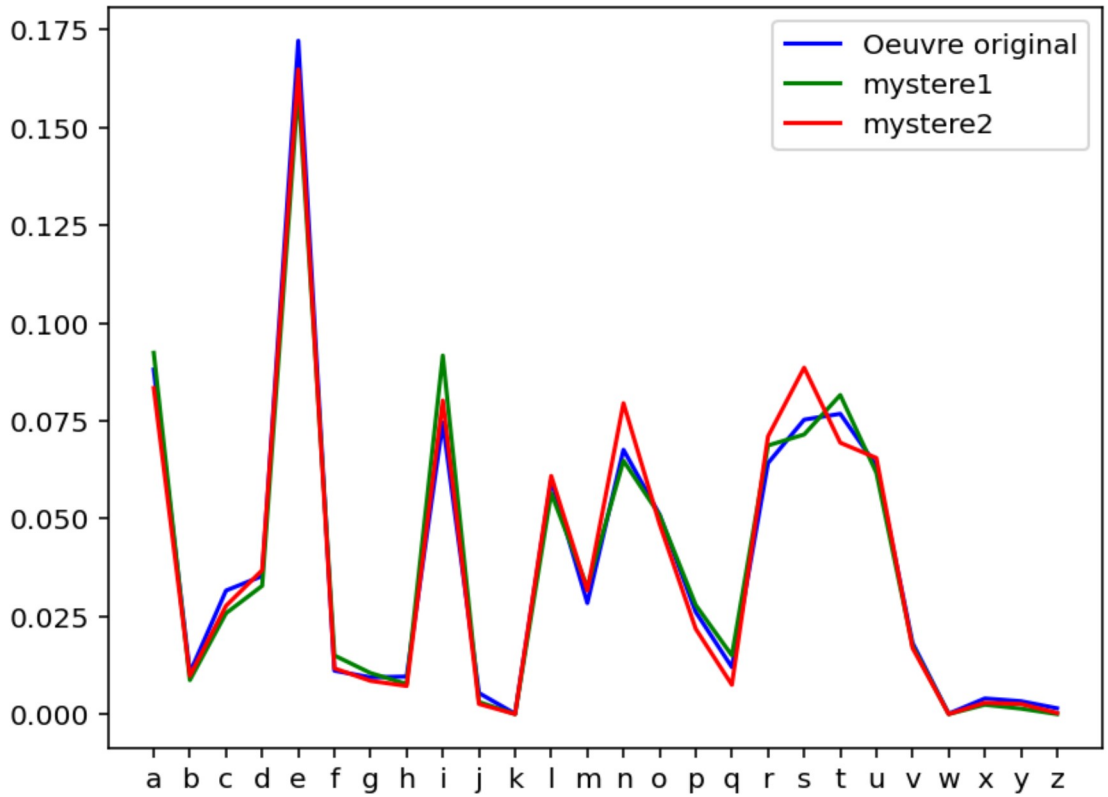
```
frequence_hugo = frequence_apparition(hugo)
frequence_mystere1_hugo = frequence_apparition(hugo_mystere1)
frequence_mystere2_hugo = frequence_apparition(hugo_mystere2)
frequence_moliere = frequence_apparition(moliere)
frequence_mystere1_moliere = frequence_apparition(moliere_mystere1)
frequence_mystere2_moliere = frequence_apparition(moliere_mystere2)
```

Au lieu d'afficher les résultats bruts, on propose de les afficher sous la forme d'un graphique :

Entrée[6]:

```
plt.figure(1)
plt.clf()
plt.plot(ALPHABET, frequence_hugo, "b", label = "Oeuvre original")
plt.plot(ALPHABET, frequence_mystere1_hugo, "g", label="mystere1")
plt.plot(ALPHABET, frequence_mystere2_hugo, "r", label="mystere2")
plt.legend()
plt.show()
plt.close()
```

Figure 1

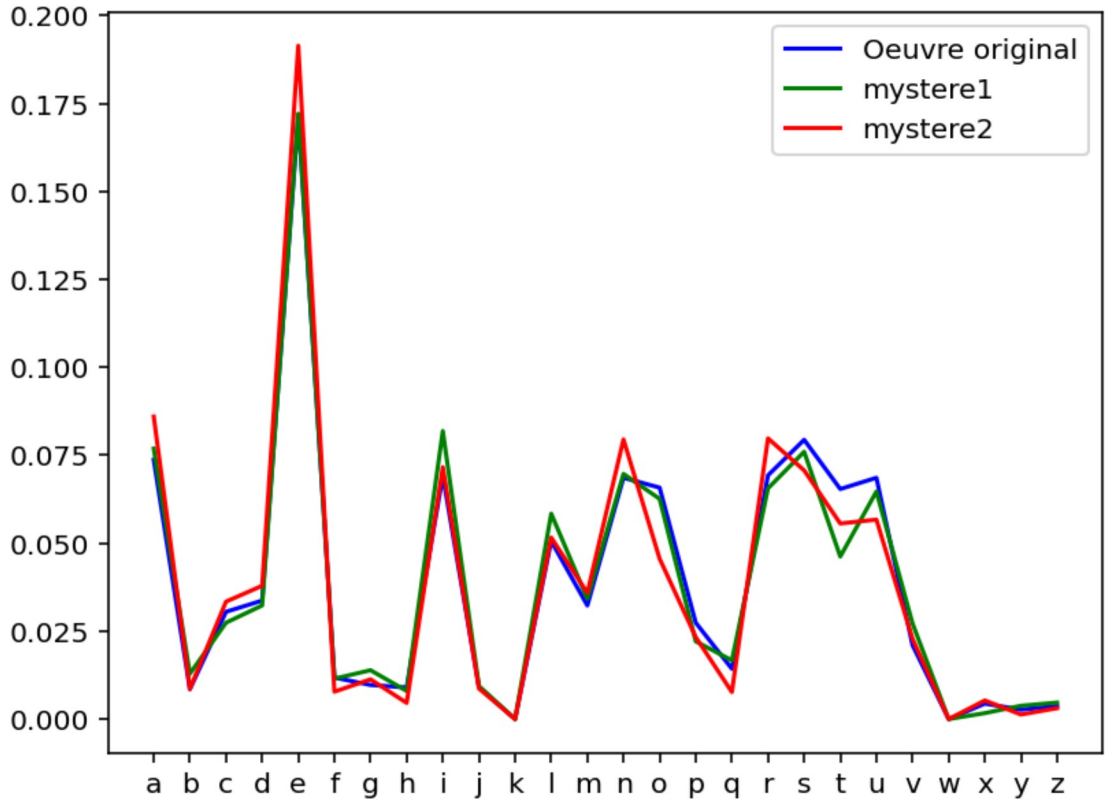



PDF
PNG
SVG

Entrée[7]:

```
plt.figure(2)
plt.clf()
plt.plot(ALPHABET, frequence_moliere, "b", label = "Oeuvre original")
plt.plot(ALPHABET, frequence_mystere1_moliere, "g", label="mystere1")
plt.plot(ALPHABET, frequence_mystere2_moliere, "r", label="mystere2")
plt.legend()
plt.show()
plt.close()
```

Figure 2

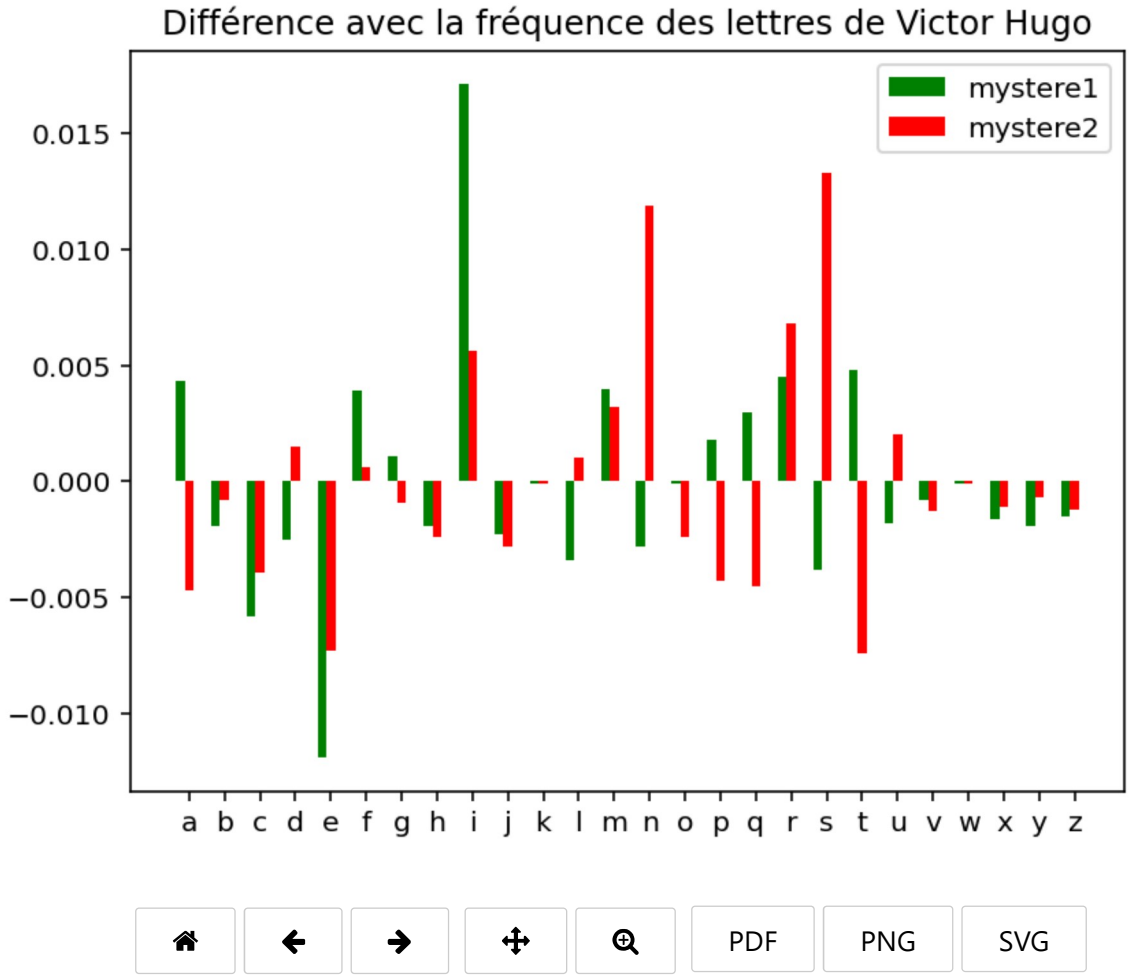


On peut voir que la courbe rouge (qui correspond à l'IA), a davantage de décrochage notamment sur l'utilisation des lettres "n", "s" et "t".
On peut, au lieu de tracer le graphique brut, proposer de tracer la différence par rapport à l'oeuvre original.

Entrée[8]:

```
# Différence des fréquences (à faire calculer aux élèves)
difference_mystere1_hugo = [frequence_mystere1_hugo[i]-frequence_hugo[i] for i in range(26)]
difference_mystere2_hugo = [frequence_mystere2_hugo[i]-frequence_hugo[i] for i in range(26)]
# représentation en histogramme (code à donner aux élèves)
plt.figure(3)
plt.clf()
fig, ax = plt.subplots()
bar_width = 0.25
ax.bar(np.arange(26), difference_mystere1_hugo, color="green", width=bar_width, label="mystere1")
ax.bar(np.arange(26)+bar_width, difference_mystere2_hugo, color="red", width=bar_width, label="mystere2")
ax.set_xticks(np.arange(26) + bar_width)
ax.set_xticklabels(ALPHABET)
ax.set_title("Différence avec la fréquence des lettres de Victor Hugo")
ax.legend()
plt.show()
plt.close()
```

Figure 4



On peut faire la somme de l'ensemble de ces fréquences pour avoir une meilleure lisibilité de ce graphique :

Entrée[15]:

```
# une phrase de conclusion
somme_diff_mystere1_hugo = sum([abs(difference_mystere1_hugo[i]) for i in range(26)])
somme_diff_mystere2_hugo = sum([abs(difference_mystere2_hugo[i]) for i in range(26)])
print(f"La somme des différences (en valeur absolue) est de {somme_diff_mystere1_hugo:0.4f} \
    pour le mystere1 et {somme_diff_mystere2_hugo:0.4f} pour le mystere2")
```

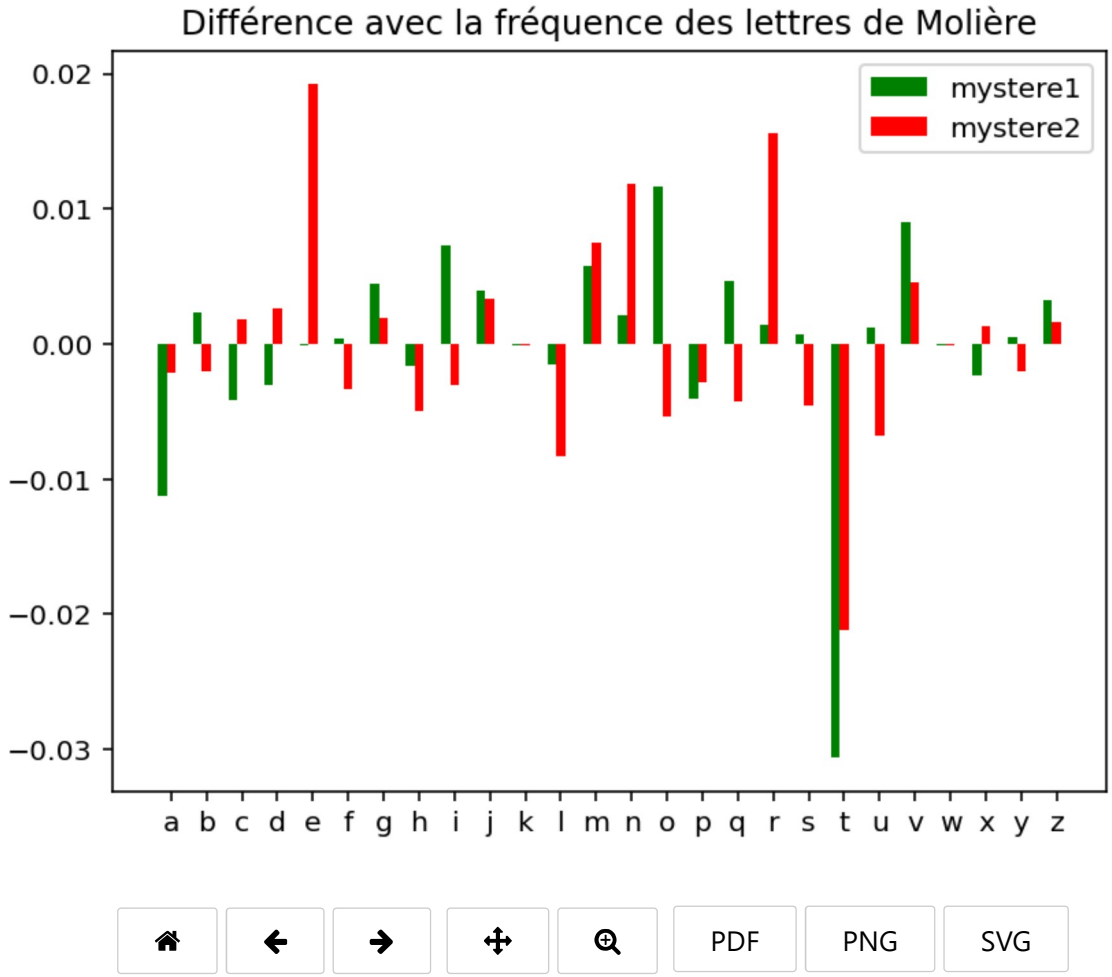
```
La somme des différences (en valeur absolue) est de 0.0887     pour le mystere1 et 0.0918 pour le mystere2
```

Les mêmes résultats pour Molière :

Entrée[16]:

```
# Différence des fréquences
difference_mystere1_moliere = [frequence_mystere1_moliere[i]-frequence_hugo[i] for i in range(26)]
difference_mystere2_moliere = [frequence_mystere2_moliere[i]-frequence_hugo[i] for i in range(26)]
# représentation en histogramme (code à donner aux élèves)
plt.figure(4)
plt.clf()
fig, ax = plt.subplots()
bar_width = 0.25
ax.bar(np.arange(26), difference_mystere1_moliere, color="green", width=bar_width, label="mystere1")
ax.bar(np.arange(26)+bar_width, difference_mystere2_moliere, color="red", width=bar_width, label="mystere2")
ax.set_xticks(np.arange(26) + bar_width)
ax.set_xticklabels(ALPHABET)
ax.set_title("Différence avec la fréquence des lettres de Molière")
ax.legend()
plt.show()
plt.close()

somme_diff_mystere1_moliere = sum([abs(difference_mystere1_moliere[i]) for i in range(26)])
somme_diff_mystere2_moliere = sum([abs(difference_mystere2_moliere[i]) for i in range(26)])
print(f"La somme des différences (en valeur absolue) est de {somme_diff_mystere1_moliere:0.4f} \
    pour le mystere1 et {somme_diff_mystere2_moliere:0.4f} pour le mystere2")

```

Figure 5




```
La somme des différences (en valeur absolue) est de 0.1174     pour le mystere1 et 0.1423 pour le mystere2
```

Conclusion :

Est-ce qu'il est possible de conclure sur quel est le texte avec cette première étape ? Cela semble compliqué, la différence ne semble pas vraiment significative...

Esprit Critique : Faire un pas de côté !

Suscitons maintenant votre esprit critique : est-ce qu'il est vraiment possible de détecter, par la fréquence des lettres, l'écriture d'une personne ?

Entrée[17]:

```
# Voici un extrait de La Comédie humaine - Volume 16. Études philosophiques et Études analytiques de Honoré Balzac
balzac = """Il doit paraître étrange que deux personnages revêtus de charges si
éminentes fissent ainsi le métier des chats. Mais pour qui fouille
les trésors historiques de ce temps, où les intérêts se croisaient
si diversement autour du trône, que l’on peut comparer la politique
intérieure de la France à un écheveau de fil brouillé, ces deux
Florentins sont de véritables chats très à leur place dans un
chéneau. Leur dévouement à la personne de la reine-mère Catherine
de Médicis qui les avait plantés à la cour de France, les obligeait
à ne reculer devant aucune des conséquences de leur intrusion. Mais
pour expliquer comment et pourquoi les deux courtisans étaient ainsi
perchés, il faut se reporter à une scène qui venait de se passer à
deux pas de cette gouttière, au Louvre, dans cette belle salle brune,
la seule peut-être qui nous reste des appartements d’Henri II, et où
les courtisans faisaient après souper leur cour aux deux reines et au
roi. A cette époque, bourgeois et grands seigneurs soupaient les uns
à six heures, les autres à sept heures; mais les raffinés soupaient
entre huit et neuf heures. Ce repas était le dîner d’aujourd’hui.
Quelques personnes croient à tort que l’étiquette a été inventée par
Louis XIV; elle procède en France de Catherine de Médicis, qui la
créa si sévère, que le connétable Anne de Montmorency eut plus de
peine à obtenir d’entrer à cheval dans la cour du Louvre qu’à obtenir
son épée; et encore! cette distinction inouïe ne fut-elle accordée
qu’à son grand âge. Un peu relâchée sous les deux premiers rois de la
maison de Bourbon, l’étiquette prit une forme orientale sous le grand
roi, car elle est venue du Bas-Empire qui la tenait de la Perse. En
1573, non-seulement peu de personnes avaient le droit d’arriver avec
leurs gens et leurs flambeaux dans la cour du Louvre, comme sous Louis
XIV les seuls ducs et pairs entraient en carrosse sous le péristyle,
mais encore les charges qui donnaient entrée après le souper dans les
appartements se comptaient. Le maréchal de Retz, alors en faction
dans sa gouttière, offrit un jour mille écus de ce temps à l’huissier
du cabinet pour pouvoir parler à Henri III, en un moment où il n’en
avait pas le droit. Quel rire excite chez un véritable historien la
vue de la cour du château de Blois, par exemple, où les dessinateurs
mettent un gentilhomme à cheval. Ainsi donc, à cette heure, il ne se
trouvait au Louvre que les personnages les plus éminents du royaume.
La reine Élisabeth d’Autriche et sa belle-mère Catherine de Médicis
étaient assises au coin gauche de la cheminée. A l’autre coin, le
roi plongé dans son fauteuil affectait une apathie autorisée par
```

```
la digestion, il avait mangé en prince qui revenait de la chasse.
Peut-être aussi voulait-il se dispenser de parler en présence de tant
de gens qui espionnaient sa pensée. Les courtisans restaient debout
et découverts au fond de la salle. Les uns causaient à voix basse;
les autres observaient le roi en attendant de lui un regard ou une
parole. Appelé par la reine-mère, celui-ci s’entretenait pendant
quelques instants avec elle. Celui-là se hasardait à dire une parole à
Charles IX, qui répondait par un signe de tête ou par un mot bref. Un
seigneur allemand, le comte de Solern, demeurait debout dans le coin
de la cheminée auprès de la petite-fille de Charles-Quint qu’il avait
accompagnée en France. Près de cette jeune reine, se tenait sur un
tabouret sa dame d’honneur, la comtesse de Fiesque, une Strozzi parente
de Catherine. La belle madame de Sauves, une descendante de Jacques
Cœur, tour à tour maîtresse du roi de Navarre, du roi de Pologne et
du duc d’Alençon, avait été invitée à souper; mais elle était debout,
son mari n’était que secrétaire d’État. Derrière ces deux dames, les
deux Gondi causaient avec elles. Eux seuls riaient dans cette morne
assemblée."""
```

Réaliser la mise en forme pour qu'il soit comparable avec les fichiers textes ouverts.

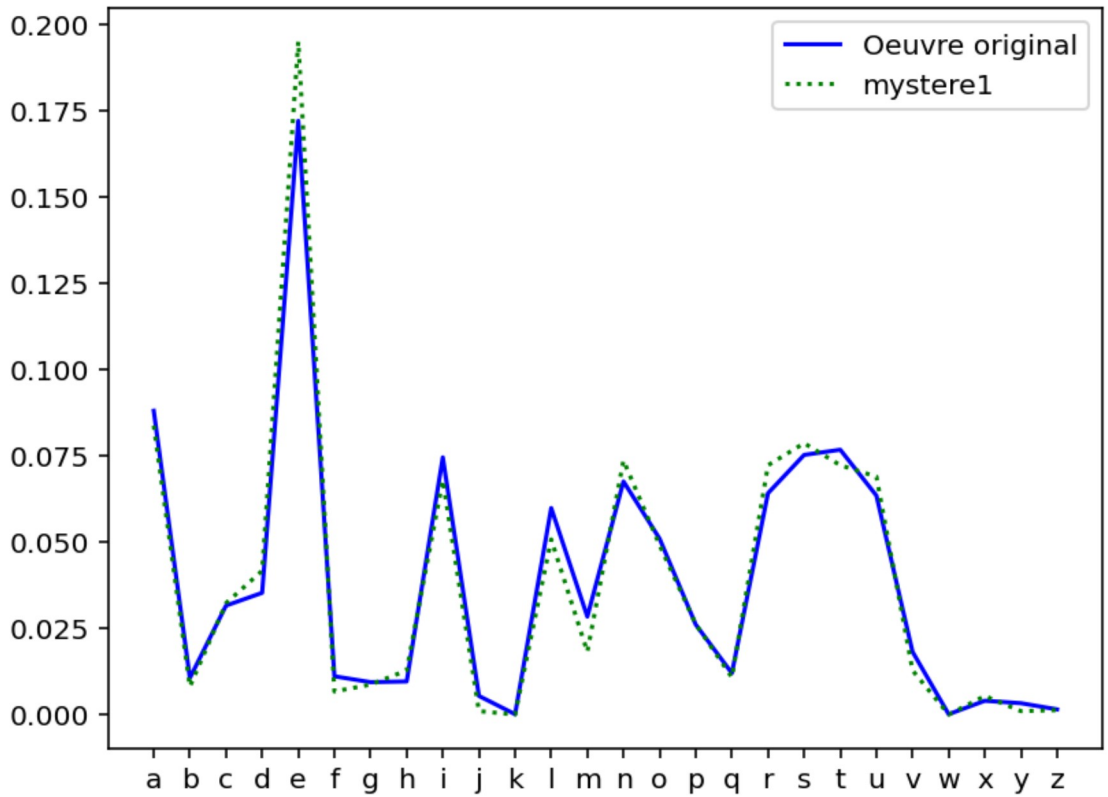
Entrée[18]:

```
# suppression des ponctuations
ponctuations = string.punctuation
for p in ponctuations:
    balzac = balzac.replace(p," ")
# gestion des accents
balzac = balzac.replace("é","e")
balzac = balzac.replace("è","e")
balzac = balzac.replace("ê","e")
balzac = balzac.replace("à","a")
balzac = balzac.replace("ù","u")
balzac = balzac.replace("ç","c")
```

Entrée[19]:

```
# Regardons les fréquences des lettres dans cet extrait :
frequence_balzac = frequence_apparition(balzac)
# Regardons le tracer correspondant entre Hugo et Balzac
plt.figure(5)
plt.clf()
plt.plot(ALPHABET, frequence_hugo, "b", label = "Oeuvre original")
plt.plot(ALPHABET, frequence_balzac, "g:", label="mystere1")

plt.legend()
plt.show()
plt.close()

difference_balzac_hugo = [frequence_balzac[i]-frequence_hugo[i] for i in range(26)]
somme_diff_balzac_hugo = sum([abs(difference_balzac_hugo[i]) for i in range(26)])
print(f"La somme des différences (en valeur absolue) est de \
    {somme_diff_balzac_hugo:0.4f} entre l'intégrale de Victor Hugo et un extrait de Balzac")

```

Figure 5




```
La somme des différences (en valeur absolue) est de     0.1158 entre l'intégrale de Victor Hugo et un extrait de Bal
zac
```

Conclusion

On peut maintenant dire que le texte de Balzac est "très proche" de celui de Victor Hugo !! Les auteurs étant contemporains, on arrive à trouver, pour cet extrait précis de Balzac en particulier (et pourtant pris au hasard) quasiment les mêmes fréquences d'apparitions. Pourtant, les deux textes n'ont pas été écris par la même personne !

Est-il alors utile de poursuivre votre lecture ?
Bien sûr, car nous allons démontrer qu'il n'est vraiment pas possible **mathématiquement parlant** de réussir à faire une quelconque différence entre un texte issu de l'auteur et un texte écrit par une IA ou un autre auteur.

Travail sur les mots

Travail sur la taille des mots

On propose de faire un premier travail sur la taille des mots : dans un premier temps sur la médiane et les quartiles. Est-ce que l'on obtient les mêmes diagramme en boîte pour les différents fichiers ?

Entrée[20]:

```
def taille_mot(texte : str) -> list:
    """A partir d'un texte, renvoie la liste des longueurs de chaque mot.
    La liste a donc la même taille que le nombre de mot dans le texte.
    """
    longueur_mot = []
    liste_mot = texte.split(" ") # création de la liste des mots en séparant le texte aux espaces
    for mot in liste_mot:
        nbre_lettre = len(mot)
        if nbre_lettre > 0 :
            longueur_mot.append(len(mot))
    return longueur_mot
```

Application :

Entrée[26]:

```
taille_hugo = taille_mot(hugo)
taille_mystere1_hugo = taille_mot(hugo_mystere1)
taille_mystere2_hugo = taille_mot(hugo_mystere2)
taille_moliere = taille_mot(moliere)
taille_mystere1_moliere = taille_mot(moliere_mystere1)
taille_mystere2_moliere = taille_mot(moliere_mystere2)

# vert = False : permet d'avoir le diagramme horizontal au lieu de verticale
# positions = [1.0,2.0,3.0] : pour passer un petit bug de programmation dans la librairie
# whis = 100 : le diagramme est tracé sur une norme non française et affiche différement les valeurs extrêmes.
plt.figure(6)
plt.clf()
plt.boxplot([taille_hugo, taille_mystere1_hugo, taille_mystere2_hugo], vert = False, positions = [1.0,2.0,3.0], wh
plt.show()
plt.close()

plt.figure(7)
plt.clf()
plt.boxplot([taille_moliere, taille_mystere1_moliere, taille_mystere2_moliere], vert = False, positions = [1.0,2.0
plt.show()
plt.close()

```

Figure 6

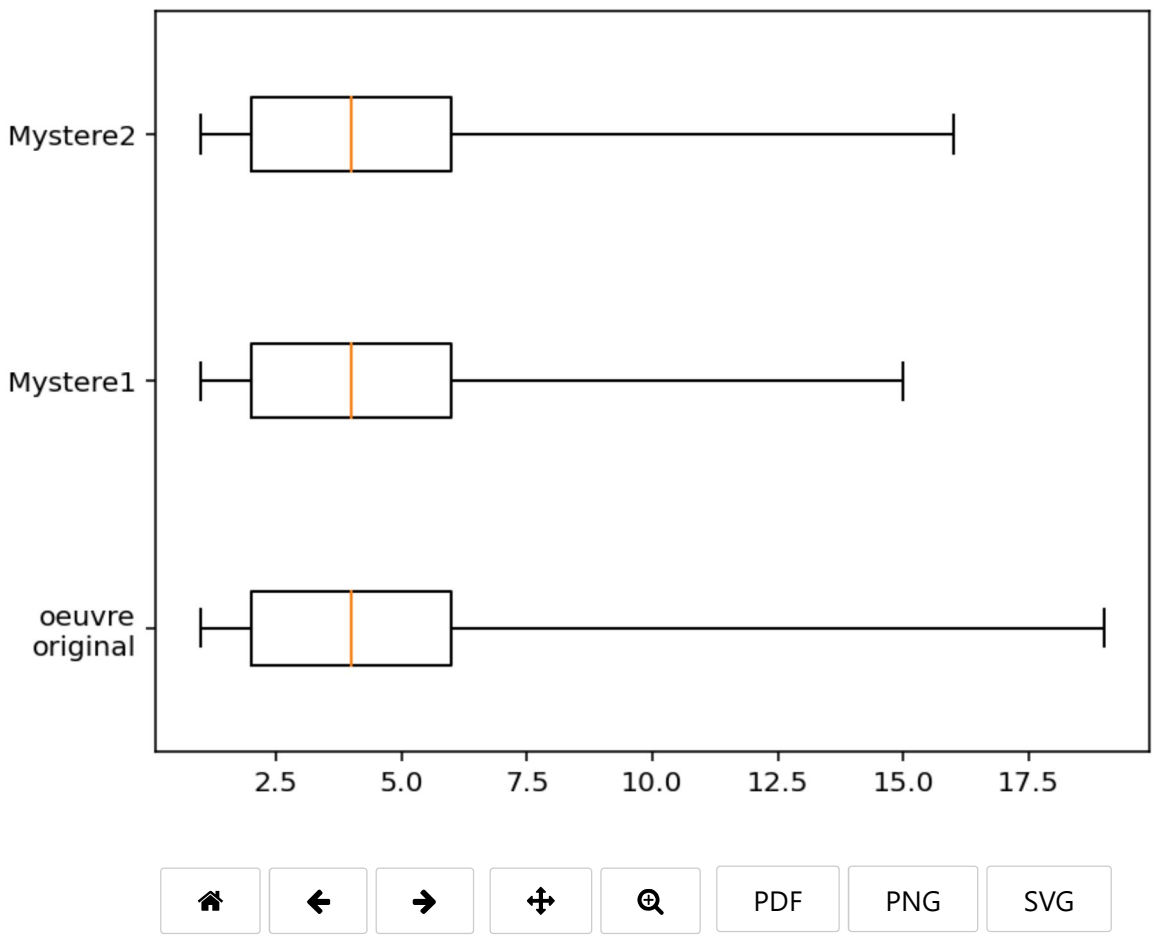
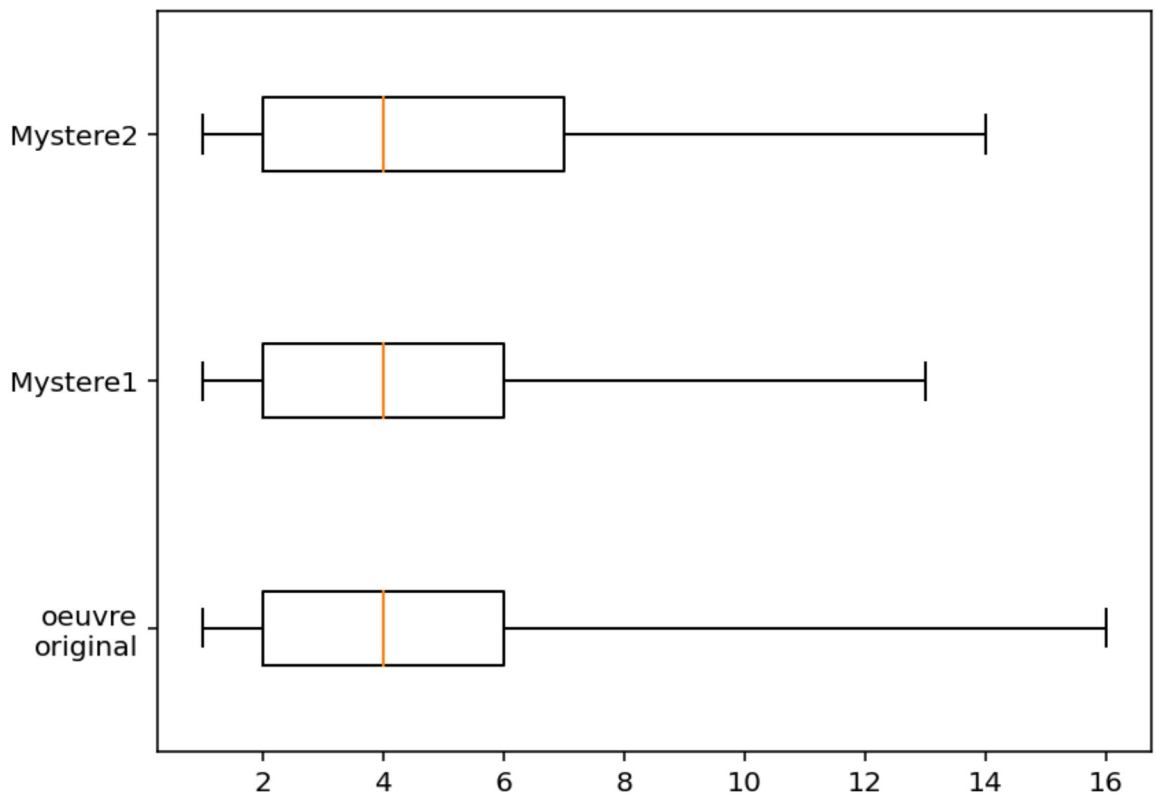

Mystere2
Mystere1
oeuvre
original
2.5
5.0
7.5
10.0
12.5
15.0
17.5
PDF
PNG
SVG

Figure 7



Les boîtes à moustache sont donc relativement proche, on n'obtient pas réellement quelque chose... On peut regarder sur le couple "moyenne / écart-type".

Entrée[22]:

```
def phrase_moyenne(liste:list)->str:
    return f"La longueur moyenne est {mean(liste):0.4f} et l'écart-type vaut {pstdev(liste):0.4f}."

print("Pour les textes de Victor Hugo : ", phrase_moyenne(taille_hugo))
print("Pour le texte Mystère 1 de Victor Hugo : ", phrase_moyenne(taille_mystere1_hugo))
print("Pour le texte Mystère 2 de Victor Hugo : ", phrase_moyenne(taille_mystere2_hugo))
print("Pour les textes de Molière : ", phrase_moyenne(taille_moliere))
print("Pour le texte Mystère 1 de Molière : ", phrase_moyenne(taille_mystere1_moliere))
print("Pour le texte Mystère 2 de Molière : ", phrase_moyenne(taille_mystere2_moliere))
```

```
Pour les textes de Victor Hugo :  La longueur moyenne est 4.4940 et l'écart-type vaut 2.6441.
Pour le texte Mystère 1 de Victor Hugo :  La longueur moyenne est 4.7099 et l'écart-type vaut 2.8018.
Pour le texte Mystère 2 de Victor Hugo :  La longueur moyenne est 4.4566 et l'écart-type vaut 2.5815.
Pour les textes de Molière :  La longueur moyenne est 4.1893 et l'écart-type vaut 2.4123.
Pour le texte Mystère 1 de Molière :  La longueur moyenne est 4.1414 et l'écart-type vaut 2.3981.
Pour le texte Mystère 2 de Molière :  La longueur moyenne est 4.6226 et l'écart-type vaut 2.5435.
```

Ainsi, mystere1, issu du texte orginal, fait moins bien que l'intelligence artificielle sur les différents couples pour Victor Hugo !

Travail sur les mots différents

On peut chercher les mots différents dans les textes générés par rapport à l'oeuvre originel.

Entrée[23]:

```
def mot_different_Vdico(texte1, texte2):
    """
    Recherche les mots présents dans le texte1 et non présents dans le texte2.
    L'occurence de chaque mot est indiqué par un dictionnaire : la clé est le mot, la valeur est le nombre d'occu
    """
    reponse = {}
    mots1 = texte1.split(" ")
    mots2 = texte2.split(" ")
    for mot in mots1:
        if mot not in mots2:
            if mot not in reponse:
                reponse[mot] = 1
            else:
                reponse[mot] = reponse[mot] + 1
    return reponse

def mot_different_Vliste(texte1, texte2):
    """
    Recherche les mots présents dans le texte1 et non présents dans le texte2.
    La fonction renvoie deux listes : la première contient les mots différents, la seconde contient leurs occuren
    """
    liste_mot = []
    liste_occurence = []
    mots1 = texte1.split(" ")
    mots2 = texte2.split(" ")
    for mot in mots1: # pour chaque mot du premier texte
        if mot not in mots2: # si ce n'est pas un mot du deuxième texte
            if mot not in liste_mot: # si ce n'est pas déjà un mot rencontré
                liste_mot.append(mot) # je l'ajoute à la liste
                liste_occurence.append(1) # je l'ai rencontré une fois
            else: # si c'est un mot déjà rencontré
                index_mot = liste_mot.index(mot) # je recherche sa place dans la liste
                liste_occurence[index_mot] = liste_occurence[index_mot] + 1 # j'indique que je le rencontre une fo
    return liste_mot, liste_occurence
```

Utilisation avec les textes de Victor Hugo :

Entrée[24]:

```
# Un exemple avec le dictionnaire en NSI
difference_mystere1_dico = mot_different_Vdico(hugo_mystere1, hugo)
print(difference_mystere1_dico)

# Exemple complet avec les listes en mathématique
difference_mystere1_liste_hugo = mot_different_Vliste(hugo_mystere1, hugo)
difference_mystere2_liste_hugo = mot_different_Vliste(hugo_mystere2, hugo)
mot1, occurence1 = difference_mystere1_liste_hugo
mot2, occurence2 = difference_mystere2_liste_hugo
print(mot2)
print(f"Pour le texte mystere1,  il y a {len(mot1)} mots différents qui sont : ")
for i in range(len(mot1)):
    print(f"  Le mot  '{mot1[i]}' est répété {occurence1[i]} fois.")
print(f"Il y a un total de {sum(occurence1)} mots différents.\n")

print(f"Pour le texte mystere2, il y a {len(mot2)} mots différents qui sont : ")
for i in range(len(mot2)):
    print(f"  Le mot  '{mot2[i]}' est répété {occurence2[i]} fois.")
print(f"Il y a un total de {sum(occurence2)} mots différents.\n")
```

```
{'i1': 2, 'tempetait': 1, 'gerontes': 1, 'd’aïeul': 1, 'n’adore': 1, 'bourrades': 1, 'gifles': 1, 'terroriste': 1, '
referais': 1, 'd’affections': 1, 's’informait': 1, 'terminaient': 1, 'inventees': 1, 'plaidaille': 1, 's’applaudissa
it': 1, 'mitigee': 1}
['vibrantes', 'vagabonde', 'reconfort', 'tourments', 'eloquents', 'errances', 'tourbillonnantes', 'vitrines', 'libra
iries', 'colorees', 'vacillante', 'alimentee', 'certitudes', 'dissolvaient', 'resonnait', 'lancinant', 'labyrinthiqu
es', 'audible', 'fugace', 'intense', 'effacaient']
Pour le texte mystere1,  il y a 16 mots différents qui sont :
  Le mot  'i1' est répété 2 fois.
  Le mot  'tempetait' est répété 1 fois.
  Le mot  'gerontes' est répété 1 fois.
  Le mot  'd’aïeul' est répété 1 fois.
  Le mot  'n’adore' est répété 1 fois.
  Le mot  'bourrades' est répété 1 fois.
  Le mot  'gifles' est répété 1 fois.
  Le mot  'terroriste' est répété 1 fois.
  Le mot  'referais' est répété 1 fois.
  Le mot  'd’affections' est répété 1 fois.
  Le mot  's’informait' est répété 1 fois.
  Le mot  'terminaient' est répété 1 fois.
  Le mot  'inventees' est répété 1 fois.
```

```
  Le mot  'plaidaille' est répété 1 fois.
  Le mot  's’applaudissait' est répété 1 fois.
  Le mot  'mitigee' est répété 1 fois.
Il y a un total de 17 mots différents.

Pour le texte mystere2, il y a 21 mots différents qui sont :
  Le mot  'vibrantes' est répété 1 fois.
  Le mot  'vagabonde' est répété 1 fois.
  Le mot  'reconfort' est répété 1 fois.
  Le mot  'tourments' est répété 1 fois.
  Le mot  'eloquents' est répété 1 fois.
  Le mot  'errances' est répété 1 fois.
  Le mot  'tourbillonnantes' est répété 1 fois.
  Le mot  'vitrines' est répété 1 fois.
  Le mot  'librairies' est répété 1 fois.
  Le mot  'colorees' est répété 1 fois.
  Le mot  'vacillante' est répété 1 fois.
  Le mot  'alimentee' est répété 1 fois.
  Le mot  'certitudes' est répété 1 fois.
  Le mot  'dissolvaient' est répété 1 fois.
  Le mot  'resonnait' est répété 1 fois.
  Le mot  'lancinant' est répété 1 fois.
  Le mot  'labyrinthiques' est répété 1 fois.
  Le mot  'audible' est répété 1 fois.
  Le mot  'fugace' est répété 1 fois.
  Le mot  'intense' est répété 1 fois.
  Le mot  'effacaient' est répété 1 fois.
Il y a un total de 21 mots différents.
```

Utilisation avec Molière :

Entrée[25]:

```
# Exemple complet avec les listes en mathématique
difference_mystere1_liste_moliere = mot_different_Vliste(moliere_mystere1, moliere)
difference_mystere2_liste_moliere = mot_different_Vliste(moliere_mystere2, moliere)
mot1, occurence1 = difference_mystere1_liste_moliere
mot2, occurence2 = difference_mystere2_liste_moliere
print(mot2)
print("Pour le texte mystere1, les mots suivants sont utilisés en plus : ")
for i in range(len(mot1)):
    print(f"  Le mot  '{mot1[i]}' est répété {occurence1[i]} fois.")
print(f"Il y a un total de {sum(occurence1)} mots différents.\n")

print("Pour le texte mystere2, les mots suivants sont utilisés en plus : ")
for i in range(len(mot2)):
    print(f"  Le mot  '{mot2[i]}' est répété {occurence2[i]} fois.")
print(f"Il y a un total de {sum(occurence2)} mots différents.\n")
```

```
['rideau', 'bûcheron', 'precipitamment', 'discutant', 'sembles', 'consultes', 'dependons', 'allongee', 'examinant',
'hmm', 'decoction', 'devrait', 'reconnaissant', 'remerciez', 'fonctionne', 'gratitude', 'medical', 'soucieux', 'dipl
ôme', 'coûtera', 'fournirai', 'paie', 'à', 'perplexe', 'arrogance', 'qualifie', 'retombee', 'desemparee', 'civiere',
'inquietez', 'herbes', 'potions', 'aleatoires', 'anxieux', 'fonctionnera', 'patients', 'horrifie', 'panique', 'prepa
ration', 'determine', 'donnerais', 'administre', 'suspense', 'recuperer', 'souriante']
Pour le texte mystere1, les mots suivants sont utilisés en plus :
  Le mot  'avancees' est répété 1 fois.
  Le mot  'avisees' est répété 1 fois.
  Le mot  'conseilleroit' est répété 1 fois.
  Le mot  'gâtera' est répété 1 fois.
  Le mot  'horloge' est répété 1 fois.
  Le mot  'abreuver' est répété 1 fois.
  Le mot  'connoîtrez' est répété 1 fois.
  Le mot  'contrefasses' est répété 1 fois.
  Le mot  'galien' est répété 2 fois.
  Le mot  'effronte' est répété 1 fois.
  Le mot  'promises' est répété 1 fois.
  Le mot  'connexite' est répété 1 fois.
  Le mot  'reculera' est répété 1 fois.
  Le mot  'carrelure' est répété 1 fois.
  Le mot  'guerira' est répété 1 fois.
  Le mot  'guerît' est répété 1 fois.
  Le mot  'vegetale' est répété 1 fois.
  Le mot  'minerale' est répété 1 fois.
```

```
  Le mot  'avortons' est répété 1 fois.
  Le mot  'salamalec' est répété 2 fois.
  Le mot  'rodrigue' est répété 1 fois.
  Le mot  'omnia' est répété 1 fois.
  Le mot  'sæcula' est répété 1 fois.
  Le mot  'sæculorum' est répété 1 fois.
  Le mot  'urine' est répété 3 fois.
  Le mot  'egrotante' est répété 1 fois.
  Le mot  'intestins' est répété 1 fois.
  Le mot  'avalez' est répété 1 fois.
  Le mot  'discerne' est répété 1 fois.
  Le mot  'pisser' est répété 4 fois.
  Le mot  'pissent' est répété 1 fois.
  Le mot  'pisse' est répété 1 fois.
  Le mot  'pisseuse' est répété 1 fois.
  Le mot  'potion' est répété 1 fois.
  Le mot  'pissatrice' est répété 1 fois.
Il y a un total de 42 mots différents.

Pour le texte mystere2, les mots suivants sont utilisés en plus :
  Le mot  'rideau' est répété 1 fois.
  Le mot  'bûcheron' est répété 1 fois.
  Le mot  'precipitamment' est répété 1 fois.
  Le mot  'discutant' est répété 1 fois.
  Le mot  'sembles' est répété 1 fois.
  Le mot  'consultes' est répété 1 fois.
  Le mot  'dependons' est répété 1 fois.
  Le mot  'allongee' est répété 2 fois.
  Le mot  'examinant' est répété 1 fois.
  Le mot  'hmm' est répété 1 fois.
  Le mot  'decoction' est répété 2 fois.
  Le mot  'devrait' est répété 1 fois.
  Le mot  'reconnaissant' est répété 1 fois.
  Le mot  'remerciez' est répété 1 fois.
  Le mot  'fonctionne' est répété 1 fois.
  Le mot  'gratitude' est répété 1 fois.
  Le mot  'medical' est répété 1 fois.
  Le mot  'soucieux' est répété 1 fois.
  Le mot  'diplôme' est répété 3 fois.
  Le mot  'coûtera' est répété 2 fois.
  Le mot  'fournirai' est répété 1 fois.
```

```
  Le mot  'paie' est répété 1 fois.
  Le mot  'à' est répété 1 fois.
  Le mot  'perplexe' est répété 1 fois.
  Le mot  'arrogance' est répété 1 fois.
  Le mot  'qualifie' est répété 1 fois.
  Le mot  'retombee' est répété 1 fois.
  Le mot  'desemparee' est répété 1 fois.
  Le mot  'civiere' est répété 1 fois.
  Le mot  'inquietez' est répété 2 fois.
  Le mot  'herbes' est répété 1 fois.
  Le mot  'potions' est répété 1 fois.
  Le mot  'aleatoires' est répété 1 fois.
  Le mot  'anxieux' est répété 1 fois.
  Le mot  'fonctionnera' est répété 1 fois.
  Le mot  'patients' est répété 1 fois.
  Le mot  'horrifie' est répété 1 fois.
  Le mot  'panique' est répété 1 fois.
  Le mot  'preparation' est répété 1 fois.
  Le mot  'determine' est répété 2 fois.
  Le mot  'donnerais' est répété 1 fois.
  Le mot  'administre' est répété 1 fois.
  Le mot  'suspense' est répété 1 fois.
  Le mot  'recuperer' est répété 1 fois.
  Le mot  'souriante' est répété 1 fois.
Il y a un total de 52 mots différents.
```

Si l'on se base sur le vocabulaire, et non sur l'étude mathématique, il apparait que mystere1 a sans doute des mots davantage susceptible d'être utilisé par Victor Hugo que mystere2.

Conclusion

Mathématiquement parlant, il est très compliqué de déterminer avec certitude quel est le texte qui provient d'une IA.

Esprit Critique : Faire un pas de côté !

Pour aller plus loin, nous allons regarder des échantillons du texte de Victor Hugo, pour nous rendre compte qu'il y a, pour un même auteur, des échantillons qui produisent des écarts très importants.

Autour de l'échantillonnage

En se basant sur une seule lettre

Nous avons aperçu un léger décrochage sur les lettres "s", "t" et "n" lors de l'étude sur la fréquence des lettres dans le texte. Nous allons donc dans un premier temps nous focaliser sur la lettre "s".
Nous allons donc :

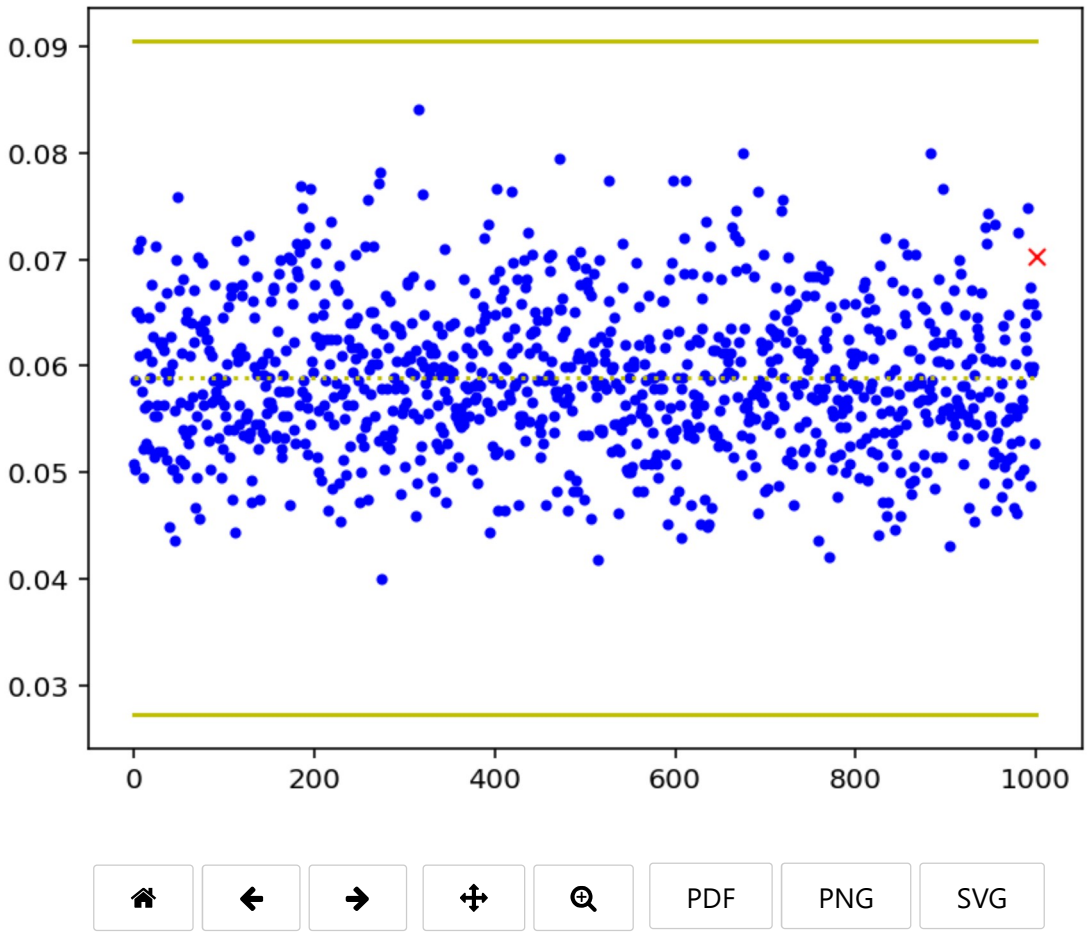
- déterminer la fréquence de la lettre "s" dans le texte intégral et dans le texte mystere2 issu de l'IA ;
- déterminer la différence d entre ces deux fréquences ;
- prendre un échantillon de même taille que mystere2 dans l'oeuvre intégrale, calculer la fréquence de la lettre "s" dans cet échantillon, regarder la différence avec la fréquence du texte intégral et comparer avec d : est-ce que c'est plus ou moins ?
- répéter l'opération pour n échantillons.

Entrée[28]:

```
from math import sqrt
from random import randint

# réécriture de la fonction frequence_apparition pour ne tenir compte que d'une seule lettre.

ALPHABET = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z

def frequence_apparition_lettre(texte:str, lettre:str) -> float:
    "Donne la fréquence de la lettre dans le texte"
    reponse = 0 # initialisation du nombre d'occurence de "lettre"
    nbre_lettres = 0 # initialisation du nombre de lettres dans le texte
    for l in texte:
        if l in ALPHABET: # je ne compte que les lettres de l'alphabet (et pas les espaces par exemple)
            if l == lettre: # si c'est la lettre que je cherche
                reponse += 1 # j'incrémente le compteur d'occurence de 1
        nbre_lettres = nbre_lettres + 1 # j'incrémente le compteur d'occurence de lettres de 1
    return reponse/nbre_lettres

# ouverture des fichiers
hugo = ouvrir_fichier_texte("Hugo_miserable.txt")
hugo_mystere2 = ouvrir_fichier_texte("Hugo_mystere2.txt")
# longueur du texte mystere 2
n = len(hugo_mystere2)

# paramétrage de l'échantillonnage et de la lettre
NBRE_ECHANTILLON = 1000
CHOIX_LETTRE = "s"

# données sur les textes complets de issus de l'IA
f_reel = frequence_apparition_lettre(hugo, CHOIX_LETTRE)
f_AI = frequence_apparition_lettre(hugo_mystere2, CHOIX_LETTRE)
d = abs(f_reel - f_AI)
nbre = 0

plt.figure(8)
plt.clf()
# Echantillonnage
for i in range(NBRE_ECHANTILLON):
    a = randint(0,len(hugo)-len(hugo_mystere2))
    f = frequence_apparition_lettre(hugo[a : a + n], CHOIX_LETTRE)
```

```
    if abs(f - f_reel) > d:
        nbre += 1
    plt.plot(i,f, "b.")

plt.plot(NBRE_ECHANTILLON + 1, f_AI, "rx")
plt.plot([0, NBRE_ECHANTILLON], [f_reel + 1/sqrt(NBRE_ECHANTILLON), f_reel + 1/sqrt(NBRE_ECHANTILLON)],'y-')
plt.plot([0, NBRE_ECHANTILLON], [f_reel - 1/sqrt(NBRE_ECHANTILLON), f_reel - 1/sqrt(NBRE_ECHANTILLON)],'y-')
plt.plot([0, NBRE_ECHANTILLON], [f_reel, f_reel],'y:')


plt.show()
plt.close()

print(f"Le nombre d'échantillon, parmi les {NBRE_ECHANTILLON} utilisé,\
qui sont plus éloigné que ce que fait l'IA est de {nbre}, soit un pourcentage de {nbre/NBRE_ECHANTILLON*100}%")
```

Figure 8




```
Le nombre d'échantillon, parmi les 1000 utilisé,qui sont plus éloigné que ce que fait l'IA est de 106, soit un pourc
entage de 10.6%
```

On obtient un résultat assez significatif : même si l'IA fait correctement la chose, la plupart des échantillons extrait du texte sont tout de même plus homogène (à 90%). Le pourcentage reste cependant assez élevé, et on voit que des échantillons font "pire". On obtient donc un résultat avec un seuil (déterminable expérimentalement avec la loi des grands nombres certainement, mais qui demande de laisser tourner le programme assez longtemps pour avoir un nombre d'échantillon satisfaisant).

Etude sur l'ensemble des lettres : les résultats sont alors moins probant.

On reprend la même idée mais en regardant la somme des distances (en valeur absolue) pour chacune des lettres.

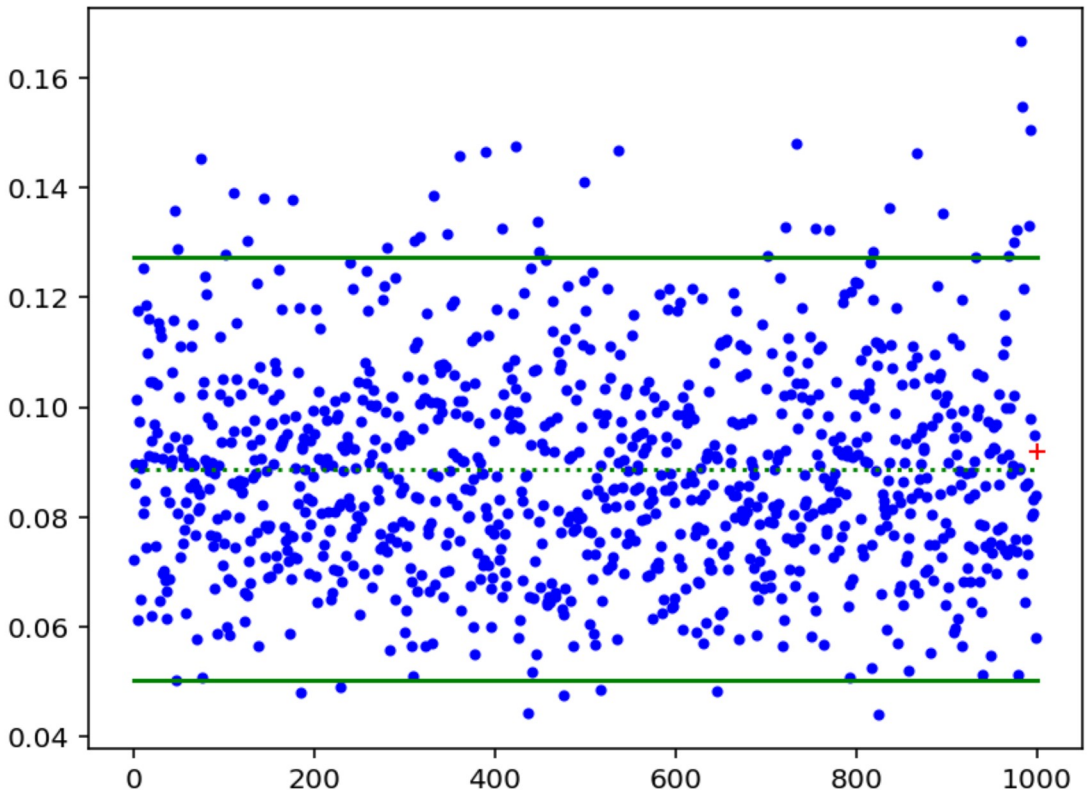
Entrée[29]:

```
from math import sqrt
from random import randint
plt.figure(9)
plt.clf()


hugo = ouvrir_fichier_texte("Hugo_miserable.txt")
hugo_mystere2 = ouvrir_fichier_texte("Hugo_mystere2.txt")
# longueur du texte mystere 2
n = len(hugo_mystere2)
# frequence d'apparition des lettres dans les oeuvres ouvertes
frequence_hugo = frequence_apparition(hugo)
frequence_mystere2 = frequence_apparition(hugo_mystere2)
distance_totale = sum([abs(frequence_hugo[i]-frequence_mystere2[i]) for i in range(26)])

# échantillonnage :
NBRE_ECHANTILLON = 1000

# Calcul des différences de fréquence entre les lettres pour des échantillons de même taille
liste_distance = []
for i in range(NBRE_ECHANTILLON):
    a = randint(0,len(hugo)-len(hugo_mystere2))
    frequence_echantillon = frequence_apparition(hugo[a:a+n])
    distance_echantillon = sum([abs(frequence_echantillon[i]-frequence_hugo[i]) for i in range(26)])
    liste_distance.append(distance_echantillon)

moyenne_echantillon = mean(liste_distance)
ecart_type = pstdev(liste_distance)
n1 = moyenne_echantillon - 2*ecart_type
n2 = moyenne_echantillon + 2*ecart_type

# Proportion des échantillons qui ne sont pas dans l'intervalle
compteur_echantillonnage = 0
compteur_pire_que_IA = 0
for i in range(NBRE_ECHANTILLON):
    if n1 < liste_distance[i] < n2:
        compteur_echantillonnage += 1
    if liste_distance[i] > distance_totale:
        compteur_pire_que_IA += 1
print(f"Le nombre d'échantillon dans l'intervalle de confiance est de {compteur_echantillonnage/NBRE_ECHANTILLON*1
```

```
print(f"Le nombre d'échantillon faisant moins bien est de {compteur_pire_que_IA/NBRE_ECHANTILLON*100}%")

# Représentation graphique
for i in range(NBRE_ECHANTILLON):
    plt.plot(i, liste_distance[i],"b.")

plt.plot(NBRE_ECHANTILLON, distance_totale,"r+")

plt.plot([0,NBRE_ECHANTILLON],[n1,n1],"g-")
plt.plot([0,NBRE_ECHANTILLON],[n2,n2],"g-")
plt.plot([0,NBRE_ECHANTILLON],[moyenne_echantillon,moyenne_echantillon],"g:")
plt.show()
plt.close()
```

```
Le nombre d'échantillon dans l'intervalle de confiance est de 95.5%
Le nombre d'échantillon faisant moins bien est de 39.900000000000006%
```

Figure 9



Conclusion d'échec :

Il y a cependant beaucoup trop (presque 40%) d'échantillon qui font moins bien que l'IA. Ce chiffre est presque stable sur plusieurs simulations : on peut en conclure que la régularité des fréquences sur l'ensemble des lettres par un auteur est donc assez volatile. Ce n'est donc pas un bon critère.

En détail lettre par lettre

On affiche uniquement la phrase donnant le nombre d'échantillons faisant "moins bien" :

- pour l'IA dans un premier temps ;
- pour un extrait pris au hasard dans un deuxième temps. On souhaite ainsi faire une comparaison des résultats. Nous vous laissons regarder chaque lettre dans le détail et nous vous proposons une moyenne.

Entrée[30]:

```
ALPHABET = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z
NBRE_ECHANTILLON = 1000
liste_IA = []
for CHOIX_LETTRE in ALPHABET:
    # données sur les textes complets de issus de l'IA
    f_reel = frequence_apparition_lettre(hugo, CHOIX_LETTRE)
    f_AI = frequence_apparition_lettre(hugo_mystere2, CHOIX_LETTRE)
    d = abs(f_reel - f_AI)
    nbre = 0
    # Echantillonnage
    for i in range(NBRE_ECHANTILLON):
        a = randint(0,len(hugo)-len(hugo_mystere2))
        f = frequence_apparition_lettre(hugo[a : a + n], CHOIX_LETTRE)
        if abs(f - f_reel) > d:
            nbre += 1
    print(f"Pour la lettre {CHOIX_LETTRE} : le pourcentage d'échantillons qui sont plus éloignés que\
    ce que fait l'IA est {nbre/NBRE_ECHANTILLON*100}%")
    liste_IA.append(nbre/NBRE_ECHANTILLON*100)
```

```
Pour la lettre a : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 65.4%
Pour la lettre b : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 74.1%
Pour la lettre c : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 37.9%
Pour la lettre d : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 63.9%
Pour la lettre e : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 60.6%
Pour la lettre f : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 73.5%
Pour la lettre g : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 72.7%
Pour la lettre h : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 28.7%
Pour la lettre i : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 33.900000000000006%
Pour la lettre j : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 34.1%
Pour la lettre k : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 13.200000000000001%
Pour la lettre l : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 76.7%
Pour la lettre m : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 45.1%
Pour la lettre n : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 1.7999999999999998%
Pour la lettre o : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 75.1%
Pour la lettre p : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 26.5%
Pour la lettre q : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 7.9%
Pour la lettre r : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 19.8%
Pour la lettre s : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 10.4%
```

Entrée[31]:

```
# Avec un échantillon pris au hasard.
ALPHABET = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z
NBRE_ECHANTILLON = 1000
liste_echantillon = []
for CHOIX_LETTRE in ALPHABET:
    # données sur les textes complets de issus de l'IA
    f_reel = frequence_apparition_lettre(hugo, CHOIX_LETTRE)
    a = randint(0,len(hugo)-len(hugo_mystere2))
    f_echantillon_precis = frequence_apparition_lettre(hugo[a : a + n], CHOIX_LETTRE)
    d = abs(f_reel - f_echantillon_precis)
    nbre = 0
    # Echantillonnage
    for i in range(NBRE_ECHANTILLON):
        a = randint(0,len(hugo)-len(hugo_mystere2))
        f = frequence_apparition_lettre(hugo[a : a + n], CHOIX_LETTRE)
        if abs(f - f_reel) > d:
            nbre += 1
    print(f"Pour la lettre {CHOIX_LETTRE} : le pourcentage d'échantillons qui sont plus éloignés que\
    cet échantillon est {nbre/NBRE_ECHANTILLON*100}%")
    liste_echantillon.append(nbre/NBRE_ECHANTILLON*100)

```

```
Pour la lettre a : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 10.4%
Pour la lettre b : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 67.10000000000001%
Pour la lettre c : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 65.4%
Pour la lettre d : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 66.9%
Pour la lettre e : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 96.0%
Pour la lettre f : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 17.299999999999997%
Pour la lettre g : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 46.7%
Pour la lettre h : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 44.9%
Pour la lettre i : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 4.3%
Pour la lettre j : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 25.1%
Pour la lettre k : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 3.4000000000000004%
Pour la lettre l : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 9.2%
Pour la lettre m : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 21.6%
Pour la lettre n : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 10.8%
Pour la lettre o : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 90.60000000000001%
Pour la lettre p : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 94.89999999999999%
Pour la lettre q : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 17.299999999999997%
Pour la lettre r : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 29.9%
```

```
Pour la lettre s : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 91.60000000000001%
Pour la lettre t : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 68.0%
Pour la lettre u : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 70.1%
Pour la lettre v : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 59.5%
Pour la lettre w : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 11.799999999999999%
Pour la lettre x : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 63.4%
Pour la lettre y : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 68.89999999999999%
Pour la lettre z : le pourcentage d'échantillons qui sont plus éloignés que ce que fait l'IA est 43.1%
```

Entrée[32]:

```
1 print(mean(liste_IA), mean(liste_echantillon))
```

```
43.65384615384615 46.08461538461538
```

Balzac et Hugo ?

Regardons maintenant ce que cela donne pour le texte de Balzac !

Entrée[34]:

```
# Avec le texte de Balzac
ALPHABET = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z
NBRE_ECHANTILLON = 1000
liste_balzac = []
for CHOIX_LETTRE in ALPHABET:
    # données sur les textes complets de issus de l'IA
    f_reel = frequence_apparition_lettre(hugo, CHOIX_LETTRE)
    a = randint(0,len(hugo)-len(balzac))
    f_echantillon_precis = frequence_apparition_lettre(hugo[a : a + n], CHOIX_LETTRE)
    d = abs(f_reel - f_echantillon_precis)
    nbre = 0
    # Echantillonnage
    for i in range(NBRE_ECHANTILLON):
        a = randint(0,len(hugo)-len(balzac))
        f = frequence_apparition_lettre(hugo[a : a + n], CHOIX_LETTRE)
        if abs(f - f_reel) > d:
            nbre += 1
    print(f"Pour la lettre {CHOIX_LETTRE} : le pourcentage d'échantillons qui sont plus éloignés\
    que celui de Balzac est {nbre/NBRE_ECHANTILLON*100}%")
    liste_balzac.append(nbre/NBRE_ECHANTILLON*100)
print(mean(liste_balzac))
```

```
Pour la lettre a : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 36.4%
Pour la lettre b : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 12.9%
Pour la lettre c : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 59.5%
Pour la lettre d : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 11.89999999999999
9%
Pour la lettre e : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 94.3999999999999
9%
Pour la lettre f : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 42.9%
Pour la lettre g : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 86.8%
Pour la lettre h : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 16.7%
Pour la lettre i : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 85.9%
Pour la lettre j : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 6.0%
Pour la lettre k : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 10.9%
Pour la lettre l : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 67.3000000000000
1%
Pour la lettre m : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 51.1%
Pour la lettre n : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 20.3%
Pour la lettre o : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 29.4%
```

```
Pour la lettre p : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 81.5%
Pour la lettre q : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 66.3%
Pour la lettre r : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 37.5%
Pour la lettre s : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 40.0%
Pour la lettre t : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 4.9%
Pour la lettre u : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 81.0%
Pour la lettre v : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 26.1%
Pour la lettre w : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 11.3%
Pour la lettre x : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 53.1%
Pour la lettre y : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 18.4%
Pour la lettre z : le pourcentage d'échantillons qui sont plus éloignés    que celui de Balzac est 41.8%
42.08846153846154
```

Conclusion

On remarque que tous les résultats sont identiques : il n'est pas du tout possible mathématiquement parlant de faire une différence quelqu'elle soit !

Entrée[ ]:
```
1
```