TP Python Séance finale

Analyse de texte

Le but du projet est d'analyser le vocabulaire utilisé dans un texte puis construire un générateur de texte "plausible". Afin d'éviter les problèmes d'accent, on utilisera un texte en anglais : https://www.irit.fr/~Philippe.Muller/alice_wonderland.utf8.txt
Mais vous pouvez aussi tester sur une texte de votre choix.

Lecture d'un fichier

Lecture complete :

```
#descripteur du fichier fichier.txt
mon_fichier = open("fichier.txt", "r")
#lecture de l'intégralité du fichier sous la
#forme d'une chaîne de caractères
contenu = mon_fichier.read()
print(contenu)
mon_fichier.close()
```

Lecture ligne par ligne :

```
mon_fichier = open("fichier.txt", "r") #descripteur du fichier fichier.txt
for ligne in mon_fichier:
      print(ligne)
mon_fichier.close() #fermeture du descripteur
```

La fonction `mon_fichier.readlines()` renvoie une liste de ligne.

Partie 1 : analyse de texte

- Faire une fonction qui charge un fichier, et qui le transforme en liste de mots (attention à la ponctuation)
- Faire une fonction qui compte le nombre d'occurence de chaque mot et qui le renvoie sous la forme d'un dictionnaire
- Faire une fonction qui renvoie les mots les plus pertinents (commencez par regarder les mots les plus fréquents)

Vous avez du remarquer entre autres problèmes, que certains mots que l'on voudrait regrouper apparaissent sous des formes différentes (pluriel des noms, verbes conjugués), et que les mots

fonctionnels (déterminants, prépositions par exemple) sont courants sans être très intéressants. Vous trouverez dans le fichier
http://www.irit.fr/~Philippe.Muller/alice_wonderland.utf8.conll
une version du texte déjà prétraité, où chaque ligne correspond à une analyse préalable d'un mot du texte (dans l'ordre du texte), avec sa forme telle qu'elle apparaît dans le texte, son lemme (cad la forme normalisée correspondant à son entrée dans le dictionnaire), et une étiquette donnant sa catégorie: nom, verbe, déterminant, etc.

- Écrivez de nouvelles fonctions pour refaire les analyses précédentes de façon plus simple avec ce fichier, en essayant de paramétrer le plus possible (prévoir de choisir les catégories à garder par exemple).

Partie 2 : n-grammes

Pour avoir des informations plus intéressantes, on peut aussi regarder les séquences de 2 mots consécutifs.

- Écrivez des fonctions pour compter toutes les séquences avec l'approche simple, garder les plus intéressantes
- Généraliser pour compter des séquences de longueur arbitraire (fixée à l'avance). On appelle ces séquences de n mots des n-grammes (bigrammes pour n=2, trigrammes pour n=3, etc).

Vous pouvez aller voir par curiosité l'inventaire historique fait par Google https://books.google.com/ngrams.

Partie 3 : Générateur aléatoire

- A partir des fréquences de bi-grammes, vous pouvez calculer la probabilité qu'un mot en suive un autre dans un texte.
- Utiliser ces probabilités pour générer un texte aléatoirement en tirant au hasard le premier mots, puis les mots suivants en tirant au hasard dans les mots les plus probables sachant le mots précédents.

Bonus :

- gérer la ponctuation
- Généraliser pour différente valeur de n (1, 3, 4, …)