But :

Les chaînes de Markov sont des processus «sans mémoire» qui vérifient la propriété de Markov. En utilisant ce concept, nous pouvons construire un générateur de texte de base, où le mot/terme suivant de notre séquence ne dépendra que du mot/terme précédent sélectionné. La transition entre ces deux termes sera basée sur nos probabilités observées à partir des données.

Si nous analysons un texte volumineux, nous pouvons utiliser des fréquences pour dériver des probabilités. Par exemple, si nous atteignons l'état «a» 100 fois dans le texte, puis 33 fois sur 100, l'état suivant pourrait être «s», car le mot « ensias » apparaît tant de fois dans le texte.

Nous allons construire un modèle pour une chaîne de Markov à partir d'un premièr fichier. L'ordre de la chaîne de Markov (c'est-à-dire le nombre de lettres dans chaque état) doit être choisi en premier.

Recherche de dataset:

On peut facilement extraire un texte d'une page html avec le module `BeautifulSoup` de python. La syntaxe est comme suite :

```
from bs4 import BeautifulSoup
from urllib.request import urlopen

html = urlopen( "https://en.wikipedia.org/wiki/Rabat" ).read( )
clean_text = ' '.join( BeautifulSoup( html, "html.parser" ).stripped_strings )
print(clean_text)
```

Vous devez effectuer un « prétraitement » de votre texte, c'est-à-dire toutes les lettres transformées en minuscules, et toute ponctuation supprimée.

Etape 1 : Modélisation du texte comme succession de lettres (Chaine de Markov d'ordre 1) :

Construisez une matrice qui se rapproche des probabilités de transition de votre article. Vous pouvez le faire en parcourant simplement toutes les paires de caractères du texte. Voici un algorithme de base qui pourrait vous aider à accomplir cela:

1- Construisez une matrice M carrée nulle de 27 lignes pour stocker vos transitions. L'entrée M [i, j] sera utilisée pour garder une trace du nombre de fois où la ième lettre de l'alphabet est suivie de la jième lettre (on considère l'espace comme une lettre).
2- Pour i de "1" à "la longueur du texte - 1", incrémentez M [x, y] si x est le ième caractère du texte et y est le (i + 1) premier caractère du texte.

3- Maintenant, la matrice M contient les décomptes de toutes les transitions. Nous voulons transformer ces décomptes en probabilités et la matrice M en matrice stochastique. Une méthode qui peut le faire. Pour chaque i de 1 à 27, additionnez les entrées sur la ième ligne, c'est-à-dire, soit : compteur [i] = M [i, 1] + M [i, 2] + M [i, 3] + ... + M [i, 27]

4- Définissez maintenant P [i, j] = M [i, j] / compteur [i] pour toutes les paires i, j. Cela donne juste une matrice de probabilités. En d'autres termes, maintenant P [i, j] est la probabilité de faire une transition de la lettre i à la lettre j.

Vous avez terminé! Vous avez maintenant une matrice de probabilités qui décrit un modèle de Markov d'ordre 1 pour votre texte!

Etape 2 : Tester la performence de votre modèle :

Testez votre modèle de Markov sur un autre texte de même nature. Comment les probabilities se comparent-elles? En guise de test, vous voudrez peut-être faire ce qui suit (ou vous pouvez créer quelque chose de plus inventif si vous le souhaitez!) :

1- Gardez une trace d'un score, à savoir, initialisez le «score» à 0.
2- Gardez une trace du nombre de «tentatives», à savoir, le nombre de fois que nous essayons de prédire la lettre suivante. Initialisez le nombre total de tentatives à 0.
3- Pour chaque i de "1" à "la longueur du deuxième texte - 1", soit x le caractère i du texte et y le (i + 1)ème caractère du texte.
4- Dans le premier texte, x est suivi de y avec une probabilité P [x, y].
5- Considérons cela comme un score, donc incrémentez le score de P [x, y] :

$$\text{score} = \text{score} + P[x, y].$$

6- Incrémentez également le nombre de tentatives :

$$\text{tentatives} = \text{tentatives} + 1.$$

7- Votre fraction globale de tentatives correctes est le score / tentatives.

Dans quelle mesure le premier texte nous dit-elle quelque chose sur le seconde texte?
Pensez à une représentation visuelle convenable du texte.

Etape 3 : Modélisation du texte avec un modèle de Markov d'ordre suppérieur :

Essayez de modéliser le première texte en utilisant une chaîne de Markov d'ordre supérieur.

Par exemple, supposons que l'ordre choisi soit fixé à 3. Une chaîne de Markov se compose alors des éléments suivants:

- Une liste de tous les 27^3 = 19683 triplets de lettres. (Remarque: cela suppose que nous rendons le texte plus uniforme en supprimant les espaces et la ponctuation supplémentaire, et que le texte n'utilise que des lettres minuscules.)
- Une matrice de probabilités de transition. Notez que, pour chaque état, il n'y a que 27 transitions possibles.

Par exemple, si la chaîne de Markov est actuellement à "ens", les états possibles qui pourraient venir ensuite sont : "ns ", "nsa", "nsb", "nsc", "nsd", ... , "nsy", "nsz"

Ainsi, la matrice des probabilités de transition devrait avoir 27^3 * 27 = 27^4 = 531441 entrées. En général, si chaque état a n lettres, alors il y a 27^n états, et la matrice des probabilités de transition a besoin de 27^(n + 1) entrées.

Vous pouvez recréer l'ensemble des étapes précédentes en utilisant ces chaînes de Markov d'ordre supérieur et voir si les performances augmentent!

Etape 4 : Modélisation du texte comme succession de mots:

Essayer de construire une matrice de transitions où les états sont les mots qui composent votre texte et comparer la performance de cette méthode avec celle de la succession des caractères.

Etape 5 : Génération de texte avec un modèle de Markov:

La plupart des exemples que vous trouverez sur le sujet commencent par un état prédéfini. Pour cela, nous utiliserons simplement notre jeton <start>.

Essayer de concaténer les états suivants en respectant les probabilités de transition de votre modèle.

Penser à utiliser les fonctions du module `random` ou du module `np.random`.

Vous devez envoyer votre compte-rendu à l'adresse mail : *m.naoum@um5r.ac.ma*

Avec comme objet « TP-Markov chain »

☆☆☆☆☆

☆☆☆

☆