Apprenstissage Automatique ou Machine Learning « Les Mots Maupassant »

Objectif écrire un programme capable de générer des phrases « cohérentes » à la façon de Maupassant.

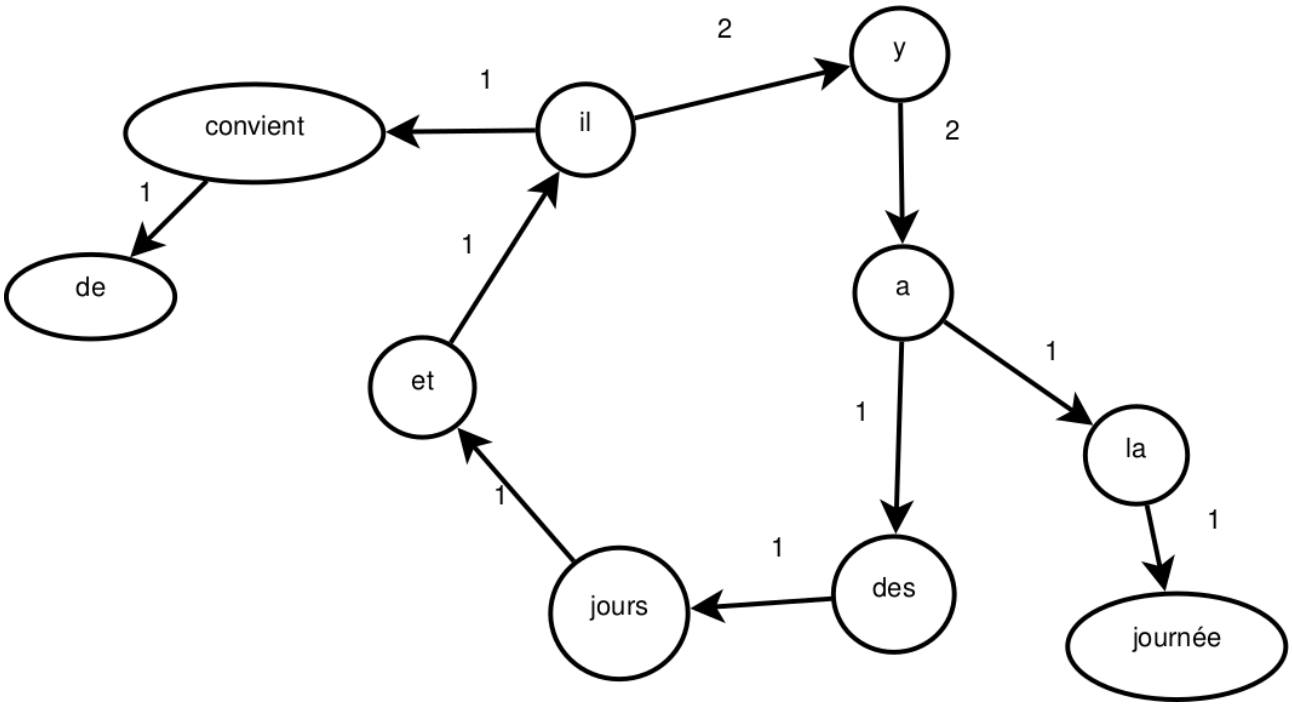
1.1 Analyser et structurer un texte « parsing »

- Ouvrir un fichier texte
- Créer un graphe orienté et pondéré de ce texte, en considérant la relation suivante :
  il existe un lien du mot A vers le mot B si et seulement si dans une phrase du texte le mot B suit le mot A. La pondération du lien mot A vers mot B est le nombre d'occurence ou le mot B suit le mot A dans de ce texte.
- La structure de données concrète pour ce graphe sera un dictionnaire de dictionnaire qui sera sauvegardé à long terme sous la forme d'un fichier JSON

Exemple :

Il y a des jours et il y a la journée NSI SNT.
Il convient de se préparer pour cette journée pleine d'échanges.

Ci-dessous un extrait d'une réprésentation du graphe suivant :



En Python cela devrait donner

```
{"il" : {"y" : 2, "convient" : 1},
 "y" : {"a" : 2},
 "a" : {"des" : 1, "la": 1},
  ...
 }
```

Attention il faudra prélablement effectuer un traitement sur les mots pour les uniformiser (ignorer certains signes de ponctuations, n'utiliser que des minusucles ...)
Après avoir générer le graphe du texte il convient de le sauvegarder de façon pérenne. Pour cela on import la bibliothèque JSON et on sauvegarde (attention à l'encodage du texte d'origine et celui du fichier JSON).

```
import json

...
...
with open('resultats.json', 'w+') as json_file: json.dump(nomdudictionnaire, json_file)
```

1.2 Exploiter le graphe pondéré

a. Importer le graphe

Dans la première partie, le texte de Maupassant a été parsé et le graphe correspondant stocké dans le fichier `resultats.json`. On va charger ce fichier pour le rendre disponible à travers un dictionnaire nommé `data`.

```
import json

with open('resultats.json', 'r') as json_file:
    data = json.load(json_file)
```

b. Marche aléatoire dans un graphe et chaînes de Markov

À partir d'un noeud du graphe (ici un mot), on doit pouvoir se rendre alétoirement au noeud suivant en respectant la pondération du graphe. Considérons par exemple le graphe pondéré du noeud "il" :

```
{'y': 4, 'convient': 1, 'faut': 2, 'regarde': 1, 'me': 2 }
```

Il suffit de tirer un nombre aléatoire `nb` entre 1 et le total des pondérations, ici 10.

Si `nb` $\leqslant 4$ le noeud suivant sera "y"

Si $4 <$ `nb` $\leqslant 5$, le noeud suivant sera "convient"

Si $5 <$ `nb` $\leqslant 7$, le noeud suivant sera "faut"

...

Si 9 `nb` le noeud suivant sera "me"

Écrire une fonction qui prend en argument un dictionnaire de ce type et retourne le mot suivant.

```
from random import randint


def next_word(dico):
    ...
    ...
    ...
    return word
```

c. Générer une phrase

Il ne reste plus qu'à générer une phrase d'une certaine longueur à partir d'un mot de départ (appelé "graine"). C'est ce que l'on appelle une marche aléatoire sur un graphe.
Il convient cependant de faire attention au cas dit « cul-de-sac ».
si le dictionnaire d'un mot est vide, on ne dispose pas de mot suivant. Dans ce cas on va faire un « saut » aléatoire vers un autre noeud (ce point pourrait être améliorer pour obtenir de "meilleures" phrases).

```
def random_sentence(nb_words, seed):
    phrase = seed.capitalize()
    ...
    ...
    for ......



    return sentence
```

d. Tests

Il ne suffit plus qu'effectuer des tests et modfier le programme en fonction des résultats obtenus.

```
random_sentence(12, 'Jeanne')
>>>"Jeanne demanda fine poussière de cuivres éclatants était un immense vestibule séparait en"
random_sentence(5, 'il')
>>> "Il ne m'étonne plus la nourriture"
```

1.3 Apprenstisage supervisé

Dans cette phase il s'agit de faire noter par un humain les phrases générées, puis que le script prenne en compte cette note en mettant en jour le graphe.
Cette phase d'entrainnement devrait avoir pour but d'améliorer les résultats.
Pour des approfondissements ou prolongements d'autres pistes peuvent être explorées. Comme par exemple écrire un bot twitter qui publira les phrases et dont le feedback de la « qualité » des phrases serait un mesure établie à partir des mentions j'aime, des réponses ou des retweets.
Cela pour être l'occasion de mettre en évidence la notion de « bias » dans le machine learning.