Ingénierie des données textuelles

De nombreuses applications utilisent des données textuelles pour faire de la prédiction : détection d'opinions, classification automatique de documents en fonction du contenu : spam - no spam, article sport vs article économie, etc...

La classification se fait de manière tout à fait classique par contre il est indispensable de traiter les documents pour pouvoir les faire interpréter par un classifieur. Le traitement des données textuelles est particulièrement difficile car il dépend des données disponibles et tout traitement n'est pas forcément justifié. Par exemple le fait de convertir tout le texte en minuscule peut faire perdre de l'information (*e.g., Mr Play indique une personne et play un verbe*), la suppression des ponctuations peut avoir des conséquences (*! est très souvent utilisé pour la détection d'opinions*), etc. En outre chaque langue possède aussi ses particularités et les librairies disponibles considèrent souvent l'anglais même s'il existe de plus en plus de ressources en différentes langues comme le français.

Le but de ce notebook est de présenter différentes approches d'ingénierie de données textuelles afin de pré-traiter les données.

Comme nous le verrons tout au cours de ce notebook, il existe de nombreuses librairies qui offrent des fonctionnalités pour pouvoir facilement traiter les données.

Installation

Avant de commencer, il est nécessaire de déjà posséder dans son environnement toutes les librairies utiles. Dans la seconde cellule nous importons toutes les librairies qui seront utiles à ce notebook. Il se peut que, lorsque vous lanciez l'éxecution de cette cellule, une soit absente. Dans ce cas il est nécessaire de l'installer. Pour cela dans la cellule suivante utiliser la commande :

! pip install nom_librairie

Attention : il est fortement conseillé lorsque l'une des librairies doit être installer de relancer le kernel de votre notebook.

Remarque : même si toutes les librairies sont importées dès le début, les librairies utiles pour des fonctions présentées au cours de ce notebook sont ré-importées de manière à indiquer d'où elles viennent et ainsi faciliter la réutilisation de la fonction dans un autre projet.

In [1]:
```
# utiliser cette cellule pour installer les librairies manquantes
# pour cela il suffit de taper dans cette cellule : !pip install no
m_librairie_manquante
# d'exécuter la cellule et de relancer la cellule suivante pour voi
r si tout se passe bien
# recommencer tant que toutes les librairies ne sont pas installées
...

# sous Colab il faut déjà intégrer ces deux librairies

!pip install langdetect
!pip install contractions
# eventuellement ne pas oublier de relancer le kernel du notebook
```

```
Collecting langdetect
  Downloading langdetect-1.0.9.tar.gz (981 kB)
     |████████████████████████████████| 981 kB 5.0 MB/s
Requirement already satisfied: six in /usr/local/lib/python3.7/dis
t-packages (from langdetect) (1.15.0)
Building wheels for collected packages: langdetect
  Building wheel for langdetect (setup.py) ... done
  Created wheel for langdetect: filename=langdetect-1.0.9-py3-none
-any.whl size=993242 sha256=3e737f991f9315288d3a76dc74f29eb99030ef
8f9bd23ba64f602d959dad9d40
  Stored in directory: /root/.cache/pip/wheels/c5/96/8a/f90c59ed25
d75e50a8c10a1b1c2d4c402e4dacfa87f3aff36a
Successfully built langdetect
Installing collected packages: langdetect
Successfully installed langdetect-1.0.9
Collecting contractions
  Downloading contractions-0.0.52-py2.py3-none-any.whl (7.2 kB)
Collecting textsearch>=0.0.21
  Downloading textsearch-0.0.21-py2.py3-none-any.whl (7.5 kB)
Collecting anyascii
  Downloading anyascii-0.3.0-py3-none-any.whl (284 kB)
     |████████████████████████████████| 284 kB 5.2 MB/s
Collecting pyahocorasick
  Downloading pyahocorasick-1.4.2.tar.gz (321 kB)
     |████████████████████████████████| 321 kB 40.5 MB/s
Building wheels for collected packages: pyahocorasick
  Building wheel for pyahocorasick (setup.py) ... done
  Created wheel for pyahocorasick: filename=pyahocorasick-1.4.2-cp
37-cp37m-linux_x86_64.whl size=85456 sha256=3e697e083aa09adbb3e75f
9b1411803452271bf90f6ea31571cfa61770e53323
  Stored in directory: /root/.cache/pip/wheels/25/19/a6/8f363d9939
162782bb8439d886469756271abc01f76fbd790f
Successfully built pyahocorasick
Installing collected packages: pyahocorasick, anyascii, textsearch
, contractions
Successfully installed anyascii-0.3.0 contractions-0.0.52 pyahocor
asick-1.4.2 textsearch-0.0.21
```

In [2]:
```
# Importation des différentes librairies utiles pour le notebook
```

```
#Sickit learn met régulièrement à jour des versions et
#indique des futurs warnings.
#ces deux lignes permettent de ne pas les afficher.
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

# librairies générales
import pickle
import pandas as pd
from scipy.stats import randint
import numpy as np
import string
import time
import base64
import re
import sys

import contractions

# librairie BeautifulSoup
from bs4 import BeautifulSoup

# librairie affichage
import matplotlib.pyplot as plt
import seaborn as sns
import wordcloud

## detection de language
import langdetect

import nltk
from nltk import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

from nltk import RegexpParser
# il est possible de charger l'ensemble des librairies en une seule
fois
# décocher le commentaire de la ligne ci-dessous
#nltk.download('all')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
from nltk import pos_tag
nltk.download('tagsets')
nltk.download("stopwords")
nltk.download('wordnet')

from nltk.corpus import stopwords

import spacy
from spacy.tokens import Span
# il faut sélectionner pour quelle langue les traitements vont être
faits.
```

```
nlp = spacy.load('en')
from spacy.lang.fr import French
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package tagsets to /root/nltk_data...
[nltk_data]   Unzipping help/tagsets.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
```

Pour pouvoir sauvegarder sur votre répertoire Google Drive, il est nécessaire de fournir une autorisation. Pour cela il suffit d'éxecuter la ligne suivante et de saisir le code donné par Google.

In [3]:
```
# pour monter son drive Google Drive local
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

Corriger éventuellement la ligne ci-dessous pour mettre le chemin vers un répertoire spécifique dans votre répertoire Google Drive :

In [4]:
```
my_local_drive='/content/gdrive/My Drive/Colab Notebooks/ML_FDS'
# Ajout du path pour les librairies, fonctions et données
sys.path.append(my_local_drive)
# Se positionner sur le répertoire associé
%cd $my_local_drive

%pwd
```

```
/content/gdrive/My Drive/Colab Notebooks/ML_FDS
```

Out[4]:
```
'/content/gdrive/My Drive/Colab Notebooks/ML_FDS'
```

In [5]:
```
# fonctions utilities (affichage, confusion, etc.)
from MyNLPUtilities import *
```

Une première analyse des documents

Très souvent pour commencer à appréhender un texte, l'une des approches consiste à déjà regarder les mots principaux d'un texte. Les word clouds offrent cette fonctionnalité. Il est également utile de connaître la langue du document. Par exemple, cela va permettre de pouvoir utiliser des librairies spécifiques, supprimer des mots inutiles pour cette langue, etc.

Il existe heureusement des librairies spécifiques comme *wordcloud* ou *langdetect*.

Nous présentons par la suite quelques premiers petits traitements pratiques qui peuvent être effectués pour nettoyer un peu les données.

In [6]:
```
import wordcloud

## detection de language
import langdetect
document = "Alice opened the door and found that it led into a smal
l passage, not much larger than a rat-hole: she knelt down and look
ed along the passage into the loveliest garden you ever saw. \
How she longed to get out of that dark hall, and wander about among
those beds of bright flowers and those cool fountains, but she coul
d not even get her head through the doorway"

# affichage des word clouds
wc = wordcloud.WordCloud(background_color='black', max_words=100,
                         max_font_size=35)
wc = wc.generate(str(document))
fig = plt.figure(num=1)
plt.axis('off')
plt.imshow(wc, cmap=None)
plt.show()


print(" Le document '", document, "' est en  ", langdetect.detect(d
ocument))
print ("la phrase il fait beau est en ", langdetect.detect("il fait
beau"))
```


```
 Le document ' Alice opened the door and found that it led into a
small passage, not much larger than a rat-hole: she knelt down and
looked along the passage into the loveliest garden you ever saw. H
ow she longed to get out of that dark hall, and wander about among
those beds of bright flowers and those cool fountains, but she cou
ld not even get her head through the doorway ' est en   en
la phrase il fait beau est en  fr
```

Encodage des données

Les données textuelles sont souvent sujettes à des problèmes d'encodage (“Latin”, “UTF8” etc). Le plus simple est de les convertir dans un format classique (UTF8).

In [7]:
```
import unicodedata
chaine = u"Klüft skräms inför på fédéral électoral große"
chaine=unicodedata.normalize('NFKD', chaine).encode('ascii','ignore
')
print (chaine)
```

```
b'Kluft skrams infor pa federal electoral groe'
```

Suppression des tags XML/HTML

Les données textuelles peuvent être issues de pages web, contenir des entêtes, etc.. L'une des premières étapes consistent à les nettoyer pour ne retenir que le texte. La librairie BeautifulSoup permet de récupérer directement le texte en supprimant les tags :
https://www.crummy.com/software/BeautifulSoup/bs4/doc/
(https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

In [8]:
```
page = """
<!DOCTYPE html>
<html> <head> <title>Machine Learning - Apprentissage</title> </hea
d>
  <body>

<h1>Le cours de Machine Learning est à a FDS </h1> (<a href=https:/
/sciences.edu.umontpellier.fr>).

 Situé à Montpellier [où il fait toujours beau]
</body> </html>"""
print (page)
```

```
<!DOCTYPE html>
<html> <head> <title>Machine Learning - Apprentissage</title> </he
ad>
  <body>

<h1>Le cours de Machine Learning est à a FDS </h1> (<a href=https:
//sciences.edu.umontpellier.fr>).

 Situé à Montpellier [où il fait toujours beau]
</body> </html>
```

In [9]:
```
from bs4 import BeautifulSoup

def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

page=strip_html (page)
print (page)
```

```
  Machine Learning - Apprentissage

Le cours de Machine Learning est à a FDS  ().

 Situé à Montpellier [où il fait toujours beau]
```

Utilisation d'expressions régulières

De nombreuses modifications peuvent être réalisées en utilisant des expressions régulières (utilisation de la librairie *re*). Par exemple la fonction suivante permet de supprimer les textes entre crochets [].

Nous verrons d'autres exemples d'expressions régulières par la suite.

In [10]:
```
import re
def remove_between_square_brackets(text):
    return re.sub('\[[^]]*\]', '', text)

page=remove_between_square_brackets(page)
print (page)
```

```
  Machine Learning - Apprentissage

Le cours de Machine Learning est à a FDS  ().

 Situé à Montpellier
```

Plus loin dans les pré-traitements des documents

La phase de pré-traitement est la phrase de préparation des données pour que ces dernières soient utilisables par un modèle d'apprentissage. Outre l'étape de nettoyage des données, il y a de nombreux pré-traitements qui peuvent ou doivent être effectués en fonction de leur type et de la tâche visée. Par exemple, l'extraction des tokens composant les phrases, la racinisation/"stemmatisation" qui vise à garder la racine des mots, la lemmatisation qui consiste à appliquer une analyse lexicale d'un texte, la suppression de mots vides ou creux (i.e., des mots qui ne sont pas discriminants pour la classification), la détection d'entité nommée, etc.

L'étape de nettoyage peut contenir elle même différentes sous-étapes selon les données (Cf. exemple précédent avec des données HTML) et la tâche visée. Elle comprend souvent la conversion des documents en minuscule et la suppression des signes de ponctuations.

Comme nous l'avons vu précédemment, il existe de nombreuses librairies pour effectuer ces différentes tâches. Dans ce notebook nous nous intéresserons plus particulièrement à :

- la librairie NLTK (Natural Language Toolkit) : http://www.nltk.org (http://www.nltk.org)
- la librairie SpaCY : https://spacy.io/ (https://spacy.io/)

Nous présentons comment ces dernières peuvent être utilisées pour réaliser les différents pré-traitements. L'importation de ces librairie se fait de la manière suivante :

In [11]:
```
import nltk
# il est possible de charger l'ensemble des librairies en une seule fois
# décocher le commentaire de la ligne ci-dessous
#nltk.download('all')

import spacy
# il faut sélectionner pour quelle langue les traitements vont être faits.
nlp = spacy.load('en')
```

In [12]:
```
import nltk
# il est possible de charger l'ensemble des librairies en une seule fois
# décocher le commentaire de la ligne ci-dessous
#nltk.download('all')

import spacy
# il faut sélectionner pour quelle langue les traitements vont être faits.
nlp = spacy.load('en')
```

Utilisation de NLTK

NLTK (Natural Language Toolkit - http://www.nltk.org (http://www.nltk.org)) est une bibliothèque Python développée par Steven Bird et Edward Loper du département d'informatique de l'université de Pennsylvanie. Elle offre de très nombreuses fonctionnalités pour manipuler les textes dans différentes langues dont le français.

L'importation de la librairie se fait par :

In [13]:
```
import nltk
# il est possible de charger l'ensemble des librairies associées en une seule fois
# pour cela décocher le commentaire de la ligne ci-dessous
#nltk.download('all')
```

In [14]:
```
document = "Alice opened the door and found that it led into a smal
l passage, not much larger than a rat-hole: she knelt down and look
ed along the passage into the loveliest garden you ever saw. \
How she longed to get out of that dark hall, and wander about among
those beds of bright flowers and those cool fountains, but she coul
d not even get her head through the doorway"
```

Sous NLTK, le découpage en phrase peut se faire à l'aide de la fonction *sent_tokenize* :

In [15]:
```
import nltk
nltk.download('punkt')
from nltk import sent_tokenize

phrases = sent_tokenize(document)
for phrase_nltk in phrases:
  print ("phrases : ",phrase_nltk)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
phrases :  Alice opened the door and found that it led into a smal
l passage, not much larger than a rat-hole: she knelt down and loo
ked along the passage into the loveliest garden you ever saw.
phrases :  How she longed to get out of that dark hall, and wander
about among those beds of bright flowers and those cool fountains,
but she could not even get her head through the doorway
```

Découpage en tokens (tokenisation)

Un texte sous python est généralement considéré comme *string*. Il est donc tout à fait possible d'utiliser les fonctions associées comme *lower* (conversion en minuscule) ou la fonction *split* associée pour découper en tokens.

In [16]:
```
print ("conversion document en minuscule")
print (document.lower())

document_splitted = document.split()
print(document_splitted)
```

```
conversion document en minuscule
alice opened the door and found that it led into a small passage,
not much larger than a rat-hole: she knelt down and looked along t
he passage into the loveliest garden you ever saw. how she longed
to get out of that dark hall, and wander about among those beds of
bright flowers and those cool fountains, but she could not even ge
t her head through the doorway
['Alice', 'opened', 'the', 'door', 'and', 'found', 'that', 'it', '
led', 'into', 'a', 'small', 'passage,', 'not', 'much', 'larger', '
than', 'a', 'rat-hole:', 'she', 'knelt', 'down', 'and', 'looked',
'along', 'the', 'passage', 'into', 'the', 'loveliest', 'garden', '
you', 'ever', 'saw.', 'How', 'she', 'longed', 'to', 'get', 'out',
'of', 'that', 'dark', 'hall,', 'and', 'wander', 'about', 'among',
'those', 'beds', 'of', 'bright', 'flowers', 'and', 'those', 'cool'
, 'fountains,', 'but', 'she', 'could', 'not', 'even', 'get', 'her'
, 'head', 'through', 'the', 'doorway']
```

Via NLTK, le découpage en tokens se fait via la fonction *word_tokenize*. Contrairement à la fonction *split* précédente, les caractères de ponctuations sont considérés comme tokens.

Remarque : comme nous pouvons le constater les ponctuations sont soient intégrées au dernier mot (*split*), soient correspondent à des tokens. Nous verrons jusque après que NTLK peut les reconnaître spécifiquement via une analyse grammaticale.

In [17]:
```
from nltk.tokenize import word_tokenize
# la liste des tokens de la première phrase
tokens = word_tokenize(phrases[0])
print(tokens)
```

```
['Alice', 'opened', 'the', 'door', 'and', 'found', 'that', 'it', '
led', 'into', 'a', 'small', 'passage', ',', 'not', 'much', 'larger
', 'than', 'a', 'rat-hole', ':', 'she', 'knelt', 'down', 'and', 'l
ooked', 'along', 'the', 'passage', 'into', 'the', 'loveliest', 'ga
rden', 'you', 'ever', 'saw', '.']
```

Etiquetage grammatical (*Part of Speech Tagging*)

L'étiquetage morpho-syntaxique (ou étiquetage grammatical) permet d'associer à chaque mot d'un texte les informations grammaticales correpondantes (e.g. verbe, préposition, ...).
Elle se fait via la fonction *pos_tag*. Elle s'applique à une phrase composée d'un ensemble de tokens et retourne les différents composants de la phrase.

In [18]:
```
from nltk import pos_tag
nltk.download('averaged_perceptron_tagger')

for phrase_nltk in phrases:
  print ("phrases : ",phrase_nltk)
  tokens = word_tokenize(phrase_nltk)
  tokens_tag = nltk.pos_tag(tokens)
  print (tokens_tag)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
phrases :  Alice opened the door and found that it led into a smal
l passage, not much larger than a rat-hole: she knelt down and loo
ked along the passage into the loveliest garden you ever saw.
[('Alice', 'NNP'), ('opened', 'VBD'), ('the', 'DT'), ('door', 'NN'
), ('and', 'CC'), ('found', 'VBD'), ('that', 'IN'), ('it', 'PRP'),
('led', 'VBD'), ('into', 'IN'), ('a', 'DT'), ('small', 'JJ'), ('pa
ssage', 'NN'), (',', ','), ('not', 'RB'), ('much', 'RB'), ('larger
', 'JJR'), ('than', 'IN'), ('a', 'DT'), ('rat-hole', 'JJ'), (':',
':'), ('she', 'PRP'), ('knelt', 'VBD'), ('down', 'RB'), ('and', 'C
C'), ('looked', 'VBD'), ('along', 'IN'), ('the', 'DT'), ('passage'
, 'NN'), ('into', 'IN'), ('the', 'DT'), ('loveliest', 'JJS'), ('ga
rden', 'NN'), ('you', 'PRP'), ('ever', 'RB'), ('saw', 'VBD'), ('.'
, '.')]
phrases :  How she longed to get out of that dark hall, and wander
about among those beds of bright flowers and those cool fountains,
but she could not even get her head through the doorway
[('How', 'WRB'), ('she', 'PRP'), ('longed', 'VBD'), ('to', 'TO'),
('get', 'VB'), ('out', 'IN'), ('of', 'IN'), ('that', 'DT'), ('dark
', 'NN'), ('hall', 'NN'), (',', ','), ('and', 'CC'), ('wander', 'V
BP'), ('about', 'IN'), ('among', 'IN'), ('those', 'DT'), ('beds',
'NNS'), ('of', 'IN'), ('bright', 'JJ'), ('flowers', 'NNS'), ('and'
, 'CC'), ('those', 'DT'), ('cool', 'JJ'), ('fountains', 'NNS'), ('
,', ','), ('but', 'CC'), ('she', 'PRP'), ('could', 'MD'), ('not',
'RB'), ('even', 'RB'), ('get', 'VB'), ('her', 'PRP$'), ('head', 'N
N'), ('through', 'IN'), ('the', 'DT'), ('doorway', 'NN')]
```

Il est possible de connaître la liste de tous les tags disponibles :

In [19]:
```
nltk.download('tagsets')
nltk.help.upenn_tagset()
```

```
[nltk_data] Downloading package tagsets to /root/nltk_data...
[nltk_data]   Package tagsets is already up-to-date!
$: dollar
    $ -$ --$ A$ C$ HK$ M$ NZ$ S$ U.S.$ US$
'': closing quotation mark
    ' ''
(: opening parenthesis
    ( [ {
): closing parenthesis
    ) ] }
,: comma
    ,
--: dash
    --
.: sentence terminator
    . ! ?
```

```
:: colon or ellipsis
    : ; ...
CC: conjunction, coordinating
    & 'n and both but either et for less minus neither nor or plus
so
    therefore times v. versus vs. whether yet
CD: numeral, cardinal
    mid-1890 nine-thirty forty-two one-tenth ten million 0.5 one f
orty-
    seven 1987 twenty '79 zero two 78-degrees eighty-four IX '60s
.025
    fifteen 271,124 dozen quintillion DM2,000 ...
DT: determiner
    all an another any both del each either every half la many muc
h nary
    neither no some such that the them these this those
EX: existential there
    there
FW: foreign word
    gemeinschaft hund ich jeux habeas Haementeria Herr K'ang-si vo
us
    lutihaw alai je jour objets salutaris fille quibusdam pas trop
Monte
    terram fiche oui corporis ...
IN: preposition or conjunction, subordinating
    astride among uppon whether out inside pro despite on by throu
ghout
    below within for towards near behind atop around if like until
below
    next into if beside ...
JJ: adjective or numeral, ordinal
    third ill-mannered pre-war regrettable oiled calamitous first
separable
    ectoplasmic battery-powered participatory fourth still-to-be-n
amed
    multilingual multi-disciplinary ...
JJR: adjective, comparative
    bleaker braver breezier briefer brighter brisker broader bumpe
r busier
    calmer cheaper choosier cleaner clearer closer colder commoner
costlier
    cozier creamier crunchier cuter ...
JJS: adjective, superlative
    calmest cheapest choicest classiest cleanest clearest closest
commonest
    corniest costliest crassest creepiest crudest cutest darkest d
eadliest
    dearest deepest densest dinkiest ...
LS: list item marker
    A A. B B. C C. D E F First G H I J K One SP-44001 SP-44002 SP-
44005
    SP-44007 Second Third Three Two * a b c d first five four one
six three
    two
```

```
MD: modal auxiliary
    can cannot could couldn't dare may might must need ought shall
should
    shouldn't will would
NN: noun, common, singular or mass
    common-carrier cabbage knuckle-duster Casino afghan shed therm
ostat
    investment slide humour falloff slick wind hyena override subh
umanity
    machinist ...
NNP: noun, proper, singular
    Motown Venneboerger Czestochwa Ranzer Conchita Trumplane Chris
tos
    Oceanside Escobar Kreisler Sawyer Cougar Yvette Ervin ODI Darr
yl CTCA
    Shannon A.K.C. Meltex Liverpool ...
NNPS: noun, proper, plural
    Americans Americas Amharas Amityvilles Amusements Anarcho-Synd
icalists
    Andalusians Andes Andruses Angels Animals Anthony Antilles Ant
iques
    Apache Apaches Apocrypha ...
NNS: noun, common, plural
    undergraduates scotches bric-a-brac products bodyguards facets
coasts
    divestitures storehouses designs clubs fragrances averages
    subjectivists apprehensions muses factory-jobs ...
PDT: pre-determiner
    all both half many quite such sure this
POS: genitive marker
    ' 's
PRP: pronoun, personal
    hers herself him himself hisself it itself me myself one onese
lf ours
    ourselves ownself self she thee theirs them themselves they th
ou thy us
PRP$: pronoun, possessive
    her his mine my our ours their thy your
RB: adverb
    occasionally unabatingly maddeningly adventurously professedly
    stirringly prominently technologically magisterially predomina
tely
    swiftly fiscally pitilessly ...
RBR: adverb, comparative
    further gloomier grander graver greater grimmer harder harsher
    healthier heavier higher however larger later leaner lengthier
less-
    perfectly lesser lonelier longer louder lower more ...
RBS: adverb, superlative
    best biggest bluntest earliest farthest first furthest hardest
    heartiest highest largest least less most nearest second tight
est worst
RP: particle
    aboard about across along apart around aside at away back befo
```

```
re behind
    by crop down ever fast for forth from go high i.e. in into jus
t later
    low more off on open out over per pie raising start teeth that
through
    under unto up up-pp upon whole with you
SYM: symbol
    % & ' '' ''. ) ). * + ,. < = > @ A[fj] U.S U.S.S.R * ** ***
TO: "to" as preposition or infinitive marker
    to
UH: interjection
    Goodbye Goody Gosh Wow Jeepers Jee-sus Hubba Hey Kee-reist Oop
s amen
    huh howdy uh dammit whammo shucks heck anyways whodunnit honey
golly
    man baby diddle hush sonuvabitch ...
VB: verb, base form
    ask assemble assess assign assume atone attention avoid bake b
alkanize
    bank begin behold believe bend benefit bevel beware bless boil
bomb
    boost brace break bring broil brush build ...
VBD: verb, past tense
    dipped pleaded swiped regummed soaked tidied convened halted r
egistered
    cushioned exacted snubbed strode aimed adopted belied figgered
    speculated wore appreciated contemplated ...
VBG: verb, present participle or gerund
    telegraphing stirring focusing angering judging stalling lacta
ting
    hankerin' alleging veering capping approaching traveling besie
ging
    encrypting interrupting erasing wincing ...
VBN: verb, past participle
    multihulled dilapidated aerosolized chaired languished paneliz
ed used
    experimented flourished imitated reunifed factored condensed s
heared
    unsettled primed dubbed desired ...
VBP: verb, present tense, not 3rd person singular
    predominate wrap resort sue twist spill cure lengthen brush te
rminate
    appear tend stray glisten obtain comprise detest tease attract
    emphasize mold postpone sever return wag ...
VBZ: verb, present tense, 3rd person singular
    bases reconstructs marks mixes displeases seals carps weaves s
natches
    slumps stretches authorizes smolders pictures emerges stockpil
es
    seduces fizzes uses bolsters slaps speaks pleads ...
WDT: WH-determiner
    that what whatever which whichever
WP: WH-pronoun
    that what whatever whatsoever which who whom whosoever
```

```
WP$: WH-pronoun, possessive
    whose
WRB: Wh-adverb
    how however whence whenever where whereby whereever wherein wh
ereof why
``: opening quotation mark
    ` ``
```

A partir de cet étiquetage, il est donc possible de ne sélectionner que des tokens correspondant à une catégorie.

In [20]:
```
word_tokens = word_tokenize(document)
pos = nltk.pos_tag(word_tokens)
selective_pos = ['NN','VBD']
selective_pos_words = []
for word,tag in pos:
     if tag in selective_pos:
         selective_pos_words.append((word,tag))
print(selective_pos_words)
```

```
[('opened', 'VBD'), ('door', 'NN'), ('found', 'VBD'), ('led', 'VBD
'), ('passage', 'NN'), ('knelt', 'VBD'), ('looked', 'VBD'), ('pass
age', 'NN'), ('garden', 'NN'), ('saw', 'VBD'), ('longed', 'VBD'),
('dark', 'NN'), ('hall', 'NN'), ('head', 'NN'), ('doorway', 'NN')]
```

Mots vides *Stop words*

Les mots vides correspondent à des mots qui sont tellement commun qu'il n'est pas nécessaire de les considérer dans l'apprentissage. NLTK possède une liste prédéfinie de mots vides faisant référence aux mots les plus courants. La première fois, il est nécessaire de télécharger les mots vides en utilisant : *nltk.download («stopwords»)*.

In [21]:
```
nltk.download("stopwords")
from nltk.corpus import stopwords
the_stopwords=set(stopwords.words("english"))
print (the_stopwords)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
{'now', "you're", 'is', 'into', 'again', 'ain', 'will', 'only', "i
sn't", "didn't", 'for', 'mustn', 'ours', 'has', 'at', 'were', 'whe
re', 'by', 'its', 'wasn', "won't", "couldn't", 'until', 'above', '
won', 'and', "don't", 'who', 'their', 'these', 're', "you'd", "has
n't", 'herself', 'through', 'during', 'it', 'so', 'about', "mustn'
t", 'more', 'himself', "wouldn't", "it's", 'hers', "she's", 'you',
'does', 'itself', 'any', 'him', 's', 'as', 'too', 'theirs', 'with'
, "haven't", 'themselves', 'a', 'are', 'own', "hadn't", 'me', 'fro
m', 'under', 'when', 'yourself', 'being', 'or', 'nor', 'between',
'most', 'doesn', 'll', 'down', 'her', 'against', 'been', 'did', 'm
y', "aren't", 'an', 'couldn', 'both', 'just', 'isn', 'those', 'how
', 'do', 'all', 'to', 'aren', "needn't", 'can', 'very', 'don', 'of
', "weren't", "that'll", 'there', 'some', 'on', "wasn't", 'no', 'h
is', 'not', 'wouldn', 't', 'o', 'hadn', "shouldn't", 'have', 'your
s', 'here', 'further', 'he', 'we', 'up', 'same', 'because', 'this'
, 'weren', 'but', 'should', 'what', 'having', 'shouldn', 'didn', '
that', 'm', "you'll", 'after', 'in', 'y', 'ma', 'i', 'off', 'am',
'over', 'whom', 'hasn', 'if', 'few', 've', "should've", 'while', '
shan', "you've", 'she', 'haven', 'which', 'mightn', 'below', "does
n't", "shan't", 'such', 'them', 'doing', 'before', 'each', 'was',
'they', 'the', 'your', 'ourselves', 'needn', 'other', 'once', 'out
', 'had', 'yourselves', 'than', 'our', 'myself', 'then', 'd', 'why
', 'be', "mightn't"}
```

Il est tout à fait possible de supprimer des stopwords de cette liste :

In [22]:
```
not_stopwords = {'this', 'd', 'o'}
#new_stopwords_list=stopwords
final_stop_words = set([word for word in the_stopwords if word not
in not_stopwords])

print (final_stop_words)
```

```
{'now', "you're", 'is', 'into', 'again', 'ain', 'will', 'only', "i
sn't", "didn't", 'for', 'mustn', 'ours', 'has', 'at', 'were', 'whe
re', 'by', 'its', 'wasn', "won't", "couldn't", 'until', 'above', '
won', 'and', "don't", 'who', 'their', 'these', 're', "you'd", "has
n't", 'herself', 'through', 'during', 'it', 'so', 'about', "mustn'
t", 'more', "wouldn't", 'himself', "it's", 'hers', "she's", 'you',
'does', 'itself', 'any', 'him', 's', 'as', 'too', 'theirs', 'with'
, "haven't", 'themselves', 'a', 'are', 'own', "hadn't", 'me', 'fro
m', 'under', 'when', 'yourself', 'being', 'or', 'nor', 'between',
'most', 'doesn', 'll', 'down', 'her', 'against', 'been', 'did', 'm
y', "aren't", 'an', 'couldn', 'both', 'just', 'isn', 'those', 'how
', 'do', 'all', 'to', 'aren', "needn't", 'can', 'very', 'don', 'of
', "weren't", "that'll", 'there', 'some', 'on', "wasn't", 'no', 'h
is', 'not', 'wouldn', 't', 'hadn', "shouldn't", 'have', 'yours', '
here', 'further', 'he', 'we', 'up', 'same', 'because', 'weren', 'b
ut', 'should', 'what', 'having', 'shouldn', 'didn', 'that', 'm', "
you'll", 'after', 'in', 'y', 'ma', 'i', 'off', 'am', 'over', 'whom
', 'hasn', 'if', 'few', 've', "should've", 'while', 'shan', "you'v
e", 'she', 'haven', 'which', 'mightn', 'below', "doesn't", "shan't
", 'such', 'them', 'doing', 'before', 'each', 'was', 'they', 'the'
, 'your', 'ourselves', 'needn', 'other', 'once', 'out', 'had', 'yo
urselves', 'than', 'our', 'myself', 'then', 'why', 'be', "mightn't
"}
```

ou d'étendre la liste des stop words.

In [23]:
```
new_stopwords=['stopword1', 'stopword2']
final_stop_words=final_stop_words.union(new_stopwords)

print (final_stop_words)
```

```
{'now', "you're", 'is', 'into', 'again', 'ain', 'will', 'only', "i
sn't", "didn't", 'for', 'mustn', 'ours', 'has', 'at', 'were', 'whe
re', 'by', 'its', 'wasn', "won't", "couldn't", 'until', 'above', '
won', 'and', "don't", 'who', 'their', 'these', 're', "you'd", "has
n't", 'herself', 'through', 'during', 'it', 'so', 'about', "mustn'
t", 'more', "wouldn't", 'himself', "it's", 'hers', "she's", 'you',
'does', 'itself', 'any', 'him', 's', 'as', 'too', 'theirs', 'with'
, "haven't", 'themselves', 'a', 'are', 'own', "hadn't", 'me', 'fro
m', 'under', 'when', 'yourself', 'being', 'or', 'nor', 'between',
'most', 'doesn', 'll', 'down', 'her', 'against', 'been', 'did', 'm
y', "aren't", 'an', 'couldn', 'both', 'just', 'isn', 'those', 'how
', 'do', 'all', 'to', 'aren', "needn't", 'can', 'very', 'don', 'of
', "weren't", "that'll", 'there', 'some', 'on', "wasn't", 'no', 'h
is', 'stopword2', 'not', 'wouldn', 't', 'hadn', "shouldn't", 'have
', 'yours', 'here', 'further', 'he', 'we', 'up', 'same', 'because'
, 'weren', 'but', 'should', 'what', 'having', 'shouldn', 'didn', '
that', 'm', "you'll", 'after', 'in', 'y', 'ma', 'i', 'off', 'am',
'over', 'whom', 'hasn', 'if', 'few', 've', "should've", 'while', '
shan', "you've", 'she', 'haven', 'which', 'stopword1', 'mightn', '
below', "doesn't", "shan't", 'such', 'them', 'doing', 'before', 'e
ach', 'was', 'they', 'the', 'your', 'ourselves', 'needn', 'other',
'once', 'out', 'had', 'yourselves', 'than', 'our', 'myself', 'then
', 'why', 'be', "mightn't"}
```

Pour supprimer les stopwords d'un document, il suffit de rechercher les tokens qui sont inclus dans les stopwords et de les supprimer.

In [24]:
```
print ("Avant suppression des stopwords")
print (word_tokens)
tokens_Alice=[word for word in word_tokens if word not in the_stopw
ords]
print ("Après suppression des stopwords")
print (tokens_Alice)
```

```
Avant suppression des stopwords
['Alice', 'opened', 'the', 'door', 'and', 'found', 'that', 'it', '
led', 'into', 'a', 'small', 'passage', ',', 'not', 'much', 'larger
', 'than', 'a', 'rat-hole', ':', 'she', 'knelt', 'down', 'and', 'l
ooked', 'along', 'the', 'passage', 'into', 'the', 'loveliest', 'ga
rden', 'you', 'ever', 'saw', '.', 'How', 'she', 'longed', 'to', 'g
et', 'out', 'of', 'that', 'dark', 'hall', ',', 'and', 'wander', 'a
bout', 'among', 'those', 'beds', 'of', 'bright', 'flowers', 'and',
'those', 'cool', 'fountains', ',', 'but', 'she', 'could', 'not', '
even', 'get', 'her', 'head', 'through', 'the', 'doorway']
Après suppression des stopwords
['Alice', 'opened', 'door', 'found', 'led', 'small', 'passage', ',
', 'much', 'larger', 'rat-hole', ':', 'knelt', 'looked', 'along',
'passage', 'loveliest', 'garden', 'ever', 'saw', '.', 'How', 'long
ed', 'get', 'dark', 'hall', ',', 'wander', 'among', 'beds', 'brigh
t', 'flowers', 'cool', 'fountains', ',', 'could', 'even', 'get', '
head', 'doorway']
```

Racinisation (*stemming*) **et lemmatisation**

NLTK utilise l'algorithme de racinisation de Porter et propose une fonction de lemmatisation. Il s'agit de deux approches différentes de transformation des flexions en leur radical ou racine. Voir https://fr.wikipedia.org/wiki/Racinisation (https://fr.wikipedia.org/wiki/Racinisation).

In [25]:
```
from nltk.stem import WordNetLemmatizer
ps=nltk.stem.porter.PorterStemmer()

print ("tokens")
print (tokens_Alice)
print("Stemming")
print([ps.stem(word) for word in tokens_Alice])

print("Lemmatisation")
lem = nltk.stem.wordnet.WordNetLemmatizer()
print([lem.lemmatize(word) for word in tokens_Alice])
```

```
tokens
['Alice', 'opened', 'door', 'found', 'led', 'small', 'passage', ',
', 'much', 'larger', 'rat-hole', ':', 'knelt', 'looked', 'along',
'passage', 'loveliest', 'garden', 'ever', 'saw', '.', 'How', 'long
ed', 'get', 'dark', 'hall', ',', 'wander', 'among', 'beds', 'brigh
t', 'flowers', 'cool', 'fountains', ',', 'could', 'even', 'get', '
head', 'doorway']
Stemming
['alic', 'open', 'door', 'found', 'led', 'small', 'passag', ',', '
much', 'larger', 'rat-hol', ':', 'knelt', 'look', 'along', 'passag
', 'loveliest', 'garden', 'ever', 'saw', '.', 'how', 'long', 'get'
, 'dark', 'hall', ',', 'wander', 'among', 'bed', 'bright', 'flower
', 'cool', 'fountain', ',', 'could', 'even', 'get', 'head', 'doorw
ay']
Lemmatisation
['Alice', 'opened', 'door', 'found', 'led', 'small', 'passage', ',
', 'much', 'larger', 'rat-hole', ':', 'knelt', 'looked', 'along',
'passage', 'loveliest', 'garden', 'ever', 'saw', '.', 'How', 'long
ed', 'get', 'dark', 'hall', ',', 'wander', 'among', 'bed', 'bright
', 'flower', 'cool', 'fountain', ',', 'could', 'even', 'get', 'hea
d', 'doorway']
```

NLTK propose aussi un stemmatiseur pour le Français :

In [26]:
```
# un autre stemmatiseur qui accepte le français
from nltk.stem.snowball import SnowballStemmer
stemmer = SnowballStemmer("french")
phrase = "malade malades maladie maladies maladive"
tokens = word_tokenize(phrase)
print ("Avant transformation \n")
print (tokens)
stemmed = [stemmer.stem(word) for word in tokens]
print ("\n Après transformation\n")
print (stemmed)
```

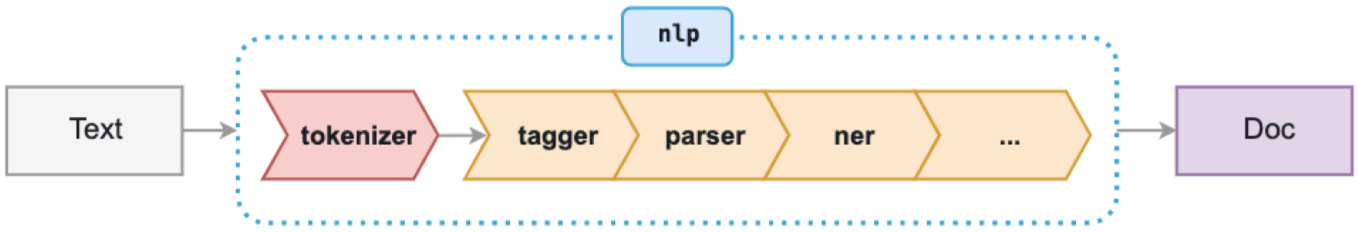
```
Avant transformation

['malade', 'malades', 'maladie', 'maladies', 'maladive']

 Après transformation

['malad', 'malad', 'malad', 'malad', 'malad']
```

Utilisation de Spacy

L'importation de la librairie se fait par :

In [27]:
```
import spacy
```

Spacy utilise un objet particulier, généralement appelé *nlp*, qui va créer un pipeline sur tous les éléments d'un document afin de générer un objet de type doc. Le pipeline de base est le suivant :



Il existe bien entendu différents modèles de pipeline en fonction de la langue.

Les ressources sont par défaut en anglais. Pour le français, il faut au préalable télécharger la ressource associée par :

In [28]:
```
!python -m spacy download fr
```

```
Collecting fr_core_news_sm==2.2.5
  Downloading https://github.com/explosion/spacy-models/releases/d
```

```
ownload/fr_core_news_sm-2.2.5/fr_core_news_sm-2.2.5.tar.gz (14.7 M
B)
     |████████████████████████████████| 14.7 MB 5.1 MB/s
Requirement already satisfied: spacy>=2.2.2 in /usr/local/lib/pyth
on3.7/dist-packages (from fr_core_news_sm==2.2.5) (2.2.4)
Requirement already satisfied: setuptools in /usr/local/lib/python
3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm==2.2.5) (57.
4.0)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/l
ib/python3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm==2.
2.5) (2.0.5)
Requirement already satisfied: thinc==7.4.0 in /usr/local/lib/pyth
on3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm==2.2.5) (7
.4.0)
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in /usr/local/l
ib/python3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm==2.
2.5) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/l
ocal/lib/python3.7/dist-packages (from spacy>=2.2.2->fr_core_news_
sm==2.2.5) (1.0.5)
Requirement already satisfied: plac<1.2.0,>=0.9.6 in /usr/local/li
b/python3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm==2.2
.5) (1.1.3)
Requirement already satisfied: blis<0.5.0,>=0.4.0 in /usr/local/li
b/python3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm==2.2
.5) (0.4.1)
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in /usr/loc
al/lib/python3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm
==2.2.5) (1.0.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/l
ib/python3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm==2.
2.5) (4.62.0)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in /usr/local/
lib/python3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm==2
.2.5) (0.8.2)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local
/lib/python3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm==
2.2.5) (3.0.5)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/loc
al/lib/python3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm
==2.2.5) (2.23.0)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/pyt
hon3.7/dist-packages (from spacy>=2.2.2->fr_core_news_sm==2.2.5) (
1.19.5)
Requirement already satisfied: importlib-metadata>=0.20 in /usr/lo
cal/lib/python3.7/dist-packages (from catalogue<1.1.0,>=0.0.7->spa
cy>=2.2.2->fr_core_news_sm==2.2.5) (4.6.4)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/lo
cal/lib/python3.7/dist-packages (from importlib-metadata>=0.20->ca
talogue<1.1.0,>=0.0.7->spacy>=2.2.2->fr_core_news_sm==2.2.5) (3.7.
4.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3
.7/dist-packages (from importlib-metadata>=0.20->catalogue<1.1.0,>
=0.0.7->spacy>=2.2.2->fr_core_news_sm==2.2.5) (3.5.0)
```

```
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/li
b/python3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy>=2.
2.2->fr_core_news_sm==2.2.5) (2021.5.30)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/pyth
on3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy>=2.2.2->f
r_core_news_sm==2.2.5) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib
/python3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy>=2.2
.2->fr_core_news_sm==2.2.5) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.
21.1 in /usr/local/lib/python3.7/dist-packages (from requests<3.0.
0,>=2.13.0->spacy>=2.2.2->fr_core_news_sm==2.2.5) (1.24.3)
Building wheels for collected packages: fr-core-news-sm
  Building wheel for fr-core-news-sm (setup.py) ... done
  Created wheel for fr-core-news-sm: filename=fr_core_news_sm-2.2.
5-py3-none-any.whl size=14727026 sha256=66f423294f2c451a215072ef45
7286bf140b6bfa0303b79b6780fd50c4c8a7ec
  Stored in directory: /tmp/pip-ephem-wheel-cache-o2lftyl3/wheels/
c9/a6/ea/0778337c34660027ee67ef3a91fb9d3600b76777a912ea1c24
Successfully built fr-core-news-sm
Installing collected packages: fr-core-news-sm
Successfully installed fr-core-news-sm-2.2.5
✔ Download and installation successful
You can now load the model via spacy.load('fr_core_news_sm')
✔ Linking successful
/usr/local/lib/python3.7/dist-packages/fr_core_news_sm -->
/usr/local/lib/python3.7/dist-packages/spacy/data/fr
You can now load the model via spacy.load('fr')
```

In [29]:

```python
from spacy.lang.fr import French

# Création d'un objet nlp
nlp = French()

# Créé en traitant une chaine de caractères avec l'objet nlp
doc = nlp("Bonjour monde !")

# Itère sur les tokens dans un Doc
for token in doc:
    print(token.text)

# remise des ressources en anglais pour la suite
nlp = spacy.load('en')
```

```
Bonjour
monde
!
```

In [30]:
```
document = "Alice opened the door and found that it led into a smal
l passage, not much larger than a rat-hole: she knelt down and look
ed along the passage into the loveliest garden you ever saw. \
How she longed to get out of that dark hall, and wander about among
those beds of bright flowers and those cool fountains, but she coul
d not even get her head through the doorway"
```

Découpage en phrases, tokenisation et analyse grammaticale

Le découpage en phrases se fait via l'attribut (*sents*) lors de la création du pipeline.

In [31]:
```
# le document est en anglais
nlp = spacy.load('en')

doc=nlp(document)
for phrases_spacy in doc.sents:
  print ("phrases : ", phrases_spacy)

#sauvegarde des phrases dans un tableau pour les manipulations ulté
rieures
sentences = [sent.string.strip() for sent in doc.sents]
```

```
phrases :  Alice opened the door and found that it led into a smal
l passage, not much larger than a rat-hole: she knelt down and loo
ked along the passage into the loveliest garden you ever saw.
phrases :  How she longed to get out of that dark hall, and wander
about among those beds of bright flowers and those cool fountains,
but she could not even get her head through the doorway
```

Spacy permet d'avoir de très nombreuses information sur les tokens : le token, l'index (associé au nombre de caractères), le lemme associé (voir plus loin), s'il s'agit d'un caractère de ponctuation, d'un espace, la forme (un X représente une majuscule et un x une minuscule), sa catégorie (Part of Speech tagging), le tag associé, s'il s'agit dune entité nommée, etc.

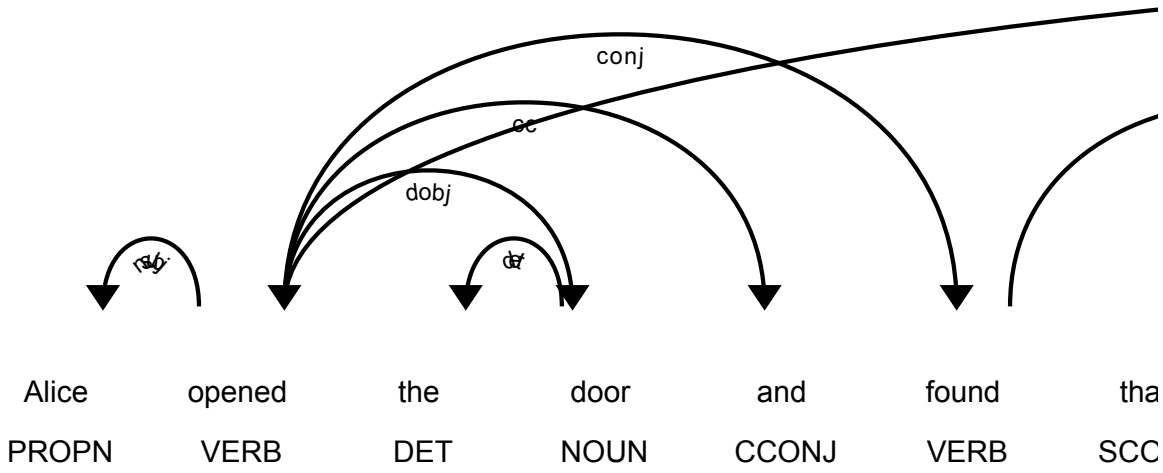
La liste des différents attributs est disponible ici : https://spacy.io/api/token (https://spacy.io/api/token)

In [32]:
```
#traitement de la première phrase
first_sentence=nlp(sentences[0])

for token in first_sentence:
    # l'objet token contient différents attributs
    print("{0}\t{1}\t{2}\t{3}\t{4}\t{5}\t{6}\t{7}\t{8}".format(
        token.text,
        token.idx,
        token.lemma_,
        token.is_punct,
        token.is_space,
        token.shape_,
        token.pos_,
        token.tag_,
        token.ent_type_
    ))
```

```
Alice    0       Alice    False   False   Xxxxx   PROPN   NNP
opened   6       open     False   False   xxxx    VERB    VBD
the      13      the      False   False   xxx     DET     DT
door     17      door     False   False   xxxx    NOUN    NN
and      22      and      False   False   xxx     CCONJ   CC
found    26      find     False   False   xxxx    VERB    VBD
that     32      that     False   False   xxxx    SCONJ   IN
it       37      -PRON-   False   False   xx      PRON    PRP
led      40      lead     False   False   xxx     VERB    VBD
into     44      into     False   False   xxxx    ADP     IN
a        49      a        False   False   x       DET     DT
small    51      small    False   False   xxxx    ADJ     JJ
passage  57      passage  False   False   xxxx    NOUN    NN
,        64      ,        True    False   ,       PUNCT   ,
not      66      not      False   False   xxx     PART    RB
much     70      much     False   False   xxxx    ADV     RB
larger   75      large    False   False   xxxx    ADJ     JJR
than     82      than     False   False   xxxx    SCONJ   IN
a        87      a        False   False   x       DET     DT
rat      89      rat      False   False   xxx     NOUN    NN
-        92      -        True    False   -       PUNCT   HYPH
hole     93      hole     False   False   xxxx    NOUN    NN
:        97      :        True    False   :       PUNCT   :
she      99      -PRON-   False   False   xxx     PRON    PRP
knelt    103     kneel    False   False   xxxx    VERB    VBD
down     109     down     False   False   xxxx    ADP     RP
and      114     and      False   False   xxx     CCONJ   CC
looked   118     look     False   False   xxxx    VERB    VBD
along    125     along    False   False   xxxx    ADP     IN
the      131     the      False   False   xxx     DET     DT
passage  135     passage  False   False   xxxx    NOUN    NN
into     143     into     False   False   xxxx    ADP     IN
the      148     the      False   False   xxx     DET     DT
loveliest        152      lovely  False   False   xxxx    ADJ
JJS
garden   162     garden   False   False   xxxx    NOUN    NN
you      169     -PRON-   False   False   xxx     PRON    PRP
ever     173     ever     False   False   xxxx    ADV     RB
saw      178     see      False   False   xxx     VERB    VBD
.        181     .        True    False   .       PUNCT   .
```

Il est également possible de visualiser les résultats de l'analyse grammaticale :

In [33]:
```
from spacy import displacy
displacy.render(first_sentence, style='dep', jupyter=True, options=
{'distance': 85})
```



Entité nommée (*name entity*)

Spacy permet également d'extraire les entités nommées d'un texte.

In [34]:
```
example_withnamedentities="Donald Trump was a President of the US a
nd now it is Joe Biden"
sentence=nlp(example_withnamedentities)
for entity in sentence.ents:
    print(entity.text + ' - ' + entity.label_ + ' - ' + str(spacy.e
xplain(entity.label_)))
```

```
Donald Trump - PERSON - People, including fictional
US - GPE - Countries, cities, states
Joe Biden - PERSON - People, including fictional
```

D'autres librairies ou traitements pratiques

Nous présentons ici différentes librairies ou traitements souvents utilisés.

In [35]:
```
#inflect est une librairie qui permet de convertir les nombres en m
ots
import inflect

phrase="They are 100"
tokens = word_tokenize(phrase)

print ("Nombre à convertir \n")
words = [word for word in tokens if word.isdigit()]
print(words)
p = inflect.engine()
numbertransf = [p.number_to_words(word) for word in tokens if word.
isdigit()]

print ("Nombre après conversion \n")
print(numbertransf)
```

```
Nombre à convertir

['100']
Nombre après conversion

['one hundred']
```

In [36]:
```
tokens = [w.lower() for w in tokens]
print (tokens)
```

```
['they', 'are', '100']
```

In [37]:
```
# Suppression de tous les termes qui ne sont pas alphanumériques
words = [word for word in tokens if word.isalpha()]
print(words)
```

```
['they', 'are']
```

In [38]:
```
import contractions

phrase="They're 100"
tokens = word_tokenize(phrase)

def replace_contractions(text):
    return contractions.fix(text)

print ("Avant remplacement\n")
print (phrase)
print ("\nAprès remplacement\n")
laphrase=replace_contractions(phrase)
print (laphrase)
```

```
Avant remplacement

They're 100

Après remplacement

they are 100
```

Les tweets ont une syntaxe très particulière et généralement les traitements se font à l'aide d'expressions régulières.

In [39]:
```
import re
tweet = '#ML is thus a good example :D ;) RT @theUser: see http://m
l.example.com'
#traitement des émoticones
emoticons_str = r"""
    (?:
        [:=;] # Eyes
        [oO\-]? # Nose (optional)
        [D\)\]\(\]/\\OpP] # Mouth
    )"""

#Prise en compte des éléments qui doivent être regroupés
regex_str = [
    emoticons_str,
    r'<[^>]+>', # HTML tags
    r'(?:@[\w_]+)', # @-mentions
    r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)", # hash-tags
    r'http[s]?://(?:[a-z]|[0-9]|[$-_@.&amp;+]|[!*\(\),]|(?:%[0-9a-f
][0-9a-f]))+', # URLs

    r'(?:(?:\d+,?)+(?:\.?\d+)?)', # nombres
    r"(?:[a-z][a-z'\-_]+[a-z])", # mots avec - et '
    r'(?:[\w_]+)', # autres mots
    r'(?:\S)' # le reste
]

tokens_re = re.compile(r'('+'|'.join(regex_str)+')', re.VERBOSE | r
e.IGNORECASE)
emoticon_re = re.compile(r'^'+emoticons_str+'$', re.VERBOSE | re.IG
NORECASE)

def tokenize(s):
    return tokens_re.findall(s)

def preprocess(s, lowercase=False):
    tokens = tokenize(s)
    if lowercase:
        tokens = [token if emoticon_re.search(token) else token.low
er() for token in tokens]
    return tokens

# un example de tweet

print ("Un exemple de tweet : \n",tweet)

print ("\nLe tweet avec un processus normal de transformation\n")
print (word_tokenize(tweet))
print ("\nLe tweet avec des expressions régulières\n")
words=preprocess(tweet)
print(words)
```

```
Un exemple de tweet :
 #ML is thus a good example :D ;) RT @theUser: see http://ml.examp
le.com

Le tweet avec un processus normal de transformation

['#', 'ML', 'is', 'thus', 'a', 'good', 'example', ':', 'D', ';', '
)', 'RT', '@', 'theUser', ':', 'see', 'http', ':', '//ml.example.c
om']

Le tweet avec des expressions régulières

['#ML', 'is', 'thus', 'a', 'good', 'example', ':D', ';)', 'RT', '@
theUser', ':', 'see', 'http://ml.example.com']
```

Une petite mise en pratique

Il est temps à présent de mettre en pratique ce que nous avons vu.

Considérez le document suivant :

In [40]:
```
testpratique=[u"""“Curiouser and curiouser!” cried Alice (she was s
o much surprised, that for the moment she quite forgot how to speak
good English); “now I’m opening out like the largest telescope that
ever was! Good-bye, feet!” (for when she looked down at her feet, t
hey seemed to be almost out of sight, they were getting so far off)
. “Oh, my poor little feet, I wonder who will put on your shoes and
stockings for you now, dears? I’m sure I shan’t be able! I shall be
a great deal too far off to trouble myself about you: you must mana
ge the best way you can;—but I must be kind to them,” thought Alice
, “or perhaps they won’t walk the way I want to go! Let me see: I’l
l give them a new pair of boots every Christmas.”
And she went on planning to herself how she would manage it. “They
must go by the carrier,” she thought; “and how funny it’ll seem, se
nding presents to one’s own feet! And how odd the directions will l
ook!
     Alice’s Right Foot, Esq.,
       Hearthrug,
         near the Fender,
           (with Alice’s love).
Oh dear, what nonsense I’m talking!”
""",u"""After a time she heard a little pattering of feet in the di
stance, and she hastily dried her eyes to see what was coming. It w
as the White Rabbit returning, splendidly dressed, with a pair of w
hite kid gloves in one hand and a large fan in the other: he came t
rotting along in a great hurry, muttering to himself as he came, “O
h! the Duchess, the Duchess! Oh! won’t she be savage if I’ve kept h
er waiting!” Alice felt so desperate that she was ready to ask help
of any one; so, when the Rabbit came near her, she began, in a low,
```

```
timid voice, “If you please, sir—” The Rabbit started violently, dr
opped the white kid gloves and the fan, and skurried away into the
darkness as hard as he could go.""",
u"""They were indeed a queer-looking party that assembled on the ba
nk—the birds with draggled feathers, the animals with their fur cli
nging close to them, and all dripping wet, cross, and uncomfortable
.
The first question of course was, how to get dry again: they had a
consultation about this, and after a few minutes it seemed quite na
tural to Alice to find herself talking familiarly with them, as if
she had known them all her life. Indeed, she had quite a long argum
ent with the Lory, who at last turned sulky, and would only say, “I
am older than you, and must know better;” and this Alice would not
allow without knowing how old it was, and, as the Lory positively r
efused to tell its age, there was no more to be said.
At last the Mouse, who seemed to be a person of authority among the
m, called out, “Sit down, all of you, and listen to me! I’ll soon m
ake you dry enough!” They all sat down at once, in a large ring, wi
th the Mouse in the middle. Alice kept her eyes anxiously fixed on
it, for she felt sure she would catch a bad cold if she did not get
dry very soon.
“Ahem!” said the Mouse with an important air, “are you all ready? T
his is the driest thing I know. Silence all round, if you please! ‘
William the Conqueror, whose cause was favoured by the pope, was so
on submitted to by the English, who wanted leaders, and had been of
late much accustomed to usurpation and conquest. Edwin and Morcar,
the earls of Mercia and Northumbria—’”
“Ugh!” said the Lory, with a shiver.
“I beg your pardon!” said the Mouse, frowning, but very politely: “
Did you speak?”
“Not I!” said the Lory hastily.
“I thought you did,” said the Mouse. “—I proceed. ‘Edwin and Morcar
, the earls of Mercia and Northumbria, declared for him: and even S
tigand, the patriotic archbishop of Canterbury, found it advisable—
’”
“Found what?” said the Duck.
“Found it,” the Mouse replied rather crossly: “of course you know w
hat ‘it’ means.”
“I know what ‘it’ means well enough, when I find a thing,” said the
Duck: “it’s generally a frog or a worm. The question is, what did t
he archbishop find?”"""]
```

Il contient différentes phrases plus ou moins longues. Vous pourrez réaliser les opérations en utilisant soit NLTK ou Spacy.

Exercice :

1. Afficher les wordclouds associés aux documents
2. Transformer les documents de telle sorte qu'ils soient en minuscule, qu'il ne possède plus de caractères spéciaux ni uniques.
3. Transformer les en tokens de manière à ce qu'ils ne contiennent plus que des tokens de type NN et VB.
4. Enfin, transformer les tokens pour n'avoir que leur racine.

In [40]:

Solution :

In [41]:
```
wc = wordcloud.WordCloud(background_color='black', max_words=100,
                         max_font_size=35)
wc = wc.generate(str(testpratique))
fig = plt.figure(num=1)
plt.axis('off')
plt.imshow(wc, cmap=None)
plt.show()

print ("Document initial ", testpratique)
# suppression des caractères spéciaux
sentence = re.sub(r'[^\w\s]',' ', str(testpratique))
# suppression de tous les caractères uniques
sentence = re.sub(r'\s+[a-zA-Z]\s+', ' ', sentence)

# tokenisation
word_tokens = word_tokenize(sentence)
print ("Premiers tokens après nettoyage des données ", word_tokens)

# ne retenir que les categories NN et VB
# sauvegarde des mots dans un tableau selective_words.
# selective_pos_words contient le mot et son tag
pos = nltk.pos_tag(word_tokens)
selective_pos = ['NN','VB']
selective_pos_words = []
selective_words=[]
for word,tag in pos:
     if tag in selective_pos:
         selective_pos_words.append((word,tag))
         selective_words.append(word)

print ("tokens conservés de catégorie NN ou VB")
print (selective_pos_words)
# racinisation


ps=nltk.stem.porter.PorterStemmer()


print("Racinisation")
print([ps.stem(word) for word in selective_words])
```

Document initial ['“Curiouser and curiouser!” cried Alice (she wa

s so much surprised, that for the moment she quite forgot how to s peak good English); “now I’m opening out like the largest telescop e that ever was! Good-bye, feet!” (for when she looked down at her feet, they seemed to be almost out of sight, they were getting so far off). “Oh, my poor little feet, I wonder who will put on your shoes and stockings for you now, dears? I’m sure I shan’t be able! I shall be a great deal too far off to trouble myself about you: y ou must manage the best way you can;—but I must be kind to them,” thought Alice, “or perhaps they won’t walk the way I want to go! L et me see: I’ll give them a new pair of boots every Christmas.”\nA nd she went on planning to herself how she would manage it. “They must go by the carrier,” she thought; “and how funny it’ll seem, s ending presents to one’s own feet! And how odd the directions will look!\n Alice’s Right Foot, Esq.,\n Hearthrug,\n near the Fender,\n (with Alice’s love).\nOh dear, what n onsense I’m talking!”\n', 'After a time she heard a little patteri ng of feet in the distance, and she hastily dried her eyes to see what was coming. It was the White Rabbit returning, splendidly dre ssed, with a pair of white kid gloves in one hand and a large fan in the other: he came trotting along in a great hurry, muttering t o himself as he came, “Oh! the Duchess, the Duchess! Oh! won’t she be savage if I’ve kept her waiting!” Alice felt so desperate that she was ready to ask help of any one; so, when the Rabbit came nea r her, she began, in a low, timid voice, “If you please, sir—” The Rabbit started violently, dropped the white kid gloves and the fan , and skurried away into the darkness as hard as he could go.', 'T hey were indeed a queer-looking party that assembled on the bank—t he birds with draggled feathers, the animals with their fur clingi ng close to them, and all dripping wet, cross, and uncomfortable.\ nThe first question of course was, how to get dry again: they had a consultation about this, and after a few minutes it seemed quite natural to Alice to find herself talking familiarly with them, as if she had known them all her life. Indeed, she had quite a long a rgument with the Lory, who at last turned sulky, and would only sa y, “I am older than you, and must know better;” and this Alice wou ld not allow without knowing how old it was, and, as the Lory posi tively refused to tell its age, there was no more to be said.\nAt last the Mouse, who seemed to be a person of authority among them, called out, “Sit down, all of you, and listen to me! I’ll soon mak e you dry enough!” They all sat down at once, in a large ring, wit h the Mouse in the middle. Alice kept her eyes anxiously fixed on it, for she felt sure she would catch a bad cold if she did not ge t dry very soon.\n“Ahem!” said the Mouse with an important air, “a re you all ready? This is the driest thing I know. Silence all rou nd, if you please! ‘William the Conqueror, whose cause was favoure d by the pope, was soon submitted to by the English, who wanted le aders, and had been of late much accustomed to usurpation and conq uest. Edwin and Morcar, the earls of Mercia and Northumbria—’”\n“U gh!” said the Lory, with a shiver.\n“I beg your pardon!” said the Mouse, frowning, but very politely: “Did you speak?”\n“Not I!” sai d the Lory hastily.\n“I thought you did,” said the Mouse. “—I proc eed. ‘Edwin and Morcar, the earls of Mercia and Northumbria, decla red for him: and even Stigand, the patriotic archbishop of Canterb ury, found it advisable—’”\n“Found what?” said the Duck.\n“Found i

```
t," the Mouse replied rather crossly: "of course you know what 'it
' means."\n"I know what 'it' means well enough, when I find a thin
g," said the Duck: "it's generally a frog or a worm. The question
is, what did the archbishop find?"']
Premiers tokens après nettoyage des données  ['Curiouser', 'and',
'curiouser', 'cried', 'Alice', 'she', 'was', 'so', 'much', 'surpri
sed', 'that', 'for', 'the', 'moment', 'she', 'quite', 'forgot', 'h
ow', 'to', 'speak', 'good', 'English', 'now', 'm', 'opening', 'out
', 'like', 'the', 'largest', 'telescope', 'that', 'ever', 'was', '
Good', 'bye', 'feet', 'for', 'when', 'she', 'looked', 'down', 'at'
, 'her', 'feet', 'they', 'seemed', 'to', 'be', 'almost', 'out', 'o
f', 'sight', 'they', 'were', 'getting', 'so', 'far', 'off', 'Oh',
'my', 'poor', 'little', 'feet', 'wonder', 'who', 'will', 'put', 'o
n', 'your', 'shoes', 'and', 'stockings', 'for', 'you', 'now', 'dea
rs', 'm', 'sure', 'shan', 'be', 'able', 'shall', 'be', 'great', 'd
eal', 'too', 'far', 'off', 'to', 'trouble', 'myself', 'about', 'yo
u', 'you', 'must', 'manage', 'the', 'best', 'way', 'you', 'can', '
but', 'must', 'be', 'kind', 'to', 'them', 'thought', 'Alice', 'or'
, 'perhaps', 'they', 'won', 'walk', 'the', 'way', 'want', 'to', 'g
o', 'Let', 'me', 'see', 'll', 'give', 'them', 'new', 'pair', 'of',
'boots', 'every', 'Christmas', 'nAnd', 'she', 'went', 'on', 'plann
ing', 'to', 'herself', 'how', 'she', 'would', 'manage', 'it', 'The
y', 'must', 'go', 'by', 'the', 'carrier', 'she', 'thought', 'and',
'how', 'funny', 'it', 'll', 'seem', 'sending', 'presents', 'to', '
one', 'own', 'feet', 'And', 'how', 'odd', 'the', 'directions', 'wi
ll', 'look', 'Alice', 'Right', 'Foot', 'Esq', 'Hearthrug', 'near',
'the', 'Fender', 'with', 'Alice', 'love', 'nOh', 'dear', 'what', '
nonsense', 'm', 'talking', 'After', 'time', 'she', 'heard', 'littl
e', 'pattering', 'of', 'feet', 'in', 'the', 'distance', 'and', 'sh
e', 'hastily', 'dried', 'her', 'eyes', 'to', 'see', 'what', 'was',
'coming', 'It', 'was', 'the', 'White', 'Rabbit', 'returning', 'spl
endidly', 'dressed', 'with', 'pair', 'of', 'white', 'kid', 'gloves
', 'in', 'one', 'hand', 'and', 'large', 'fan', 'in', 'the', 'other
', 'he', 'came', 'trotting', 'along', 'in', 'great', 'hurry', 'mut
tering', 'to', 'himself', 'as', 'he', 'came', 'Oh', 'the', 'Duches
s', 'the', 'Duchess', 'Oh', 'won', 'she', 'be', 'savage', 'if', 'v
e', 'kept', 'her', 'waiting', 'Alice', 'felt', 'so', 'desperate',
'that', 'she', 'was', 'ready', 'to', 'ask', 'help', 'of', 'any', '
one', 'so', 'when', 'the', 'Rabbit', 'came', 'near', 'her', 'she',
'began', 'in', 'low', 'timid', 'voice', 'If', 'you', 'please', 'si
r', 'The', 'Rabbit', 'started', 'violently', 'dropped', 'the', 'wh
ite', 'kid', 'gloves', 'and', 'the', 'fan', 'and', 'skurried', 'aw
ay', 'into', 'the', 'darkness', 'as', 'hard', 'as', 'he', 'could',
'go', 'They', 'were', 'indeed', 'queer', 'looking', 'party', 'that
', 'assembled', 'on', 'the', 'bank', 'the', 'birds', 'with', 'drag
gled', 'feathers', 'the', 'animals', 'with', 'their', 'fur', 'clin
ging', 'close', 'to', 'them', 'and', 'all', 'dripping', 'wet', 'cr
oss', 'and', 'uncomfortable', 'nThe', 'first', 'question', 'of', '
course', 'was', 'how', 'to', 'get', 'dry', 'again', 'they', 'had',
'consultation', 'about', 'this', 'and', 'after', 'few', 'minutes',
'it', 'seemed', 'quite', 'natural', 'to', 'Alice', 'to', 'find', '
herself', 'talking', 'familiarly', 'with', 'them', 'as', 'if', 'sh
e', 'had', 'known', 'them', 'all', 'her', 'life', 'Indeed', 'she',
```

```
'had', 'quite', 'long', 'argument', 'with', 'the', 'Lory', 'who',
'at', 'last', 'turned', 'sulky', 'and', 'would', 'only', 'say', 'a
m', 'older', 'than', 'you', 'and', 'must', 'know', 'better', 'and'
, 'this', 'Alice', 'would', 'not', 'allow', 'without', 'knowing',
'how', 'old', 'it', 'was', 'and', 'as', 'the', 'Lory', 'positively
', 'refused', 'to', 'tell', 'its', 'age', 'there', 'was', 'no', 'm
ore', 'to', 'be', 'said', 'nAt', 'last', 'the', 'Mouse', 'who', 's
eemed', 'to', 'be', 'person', 'of', 'authority', 'among', 'them',
'called', 'out', 'Sit', 'down', 'all', 'of', 'you', 'and', 'listen
', 'to', 'me', 'll', 'soon', 'make', 'you', 'dry', 'enough', 'They
', 'all', 'sat', 'down', 'at', 'once', 'in', 'large', 'ring', 'wit
h', 'the', 'Mouse', 'in', 'the', 'middle', 'Alice', 'kept', 'her',
'eyes', 'anxiously', 'fixed', 'on', 'it', 'for', 'she', 'felt', 's
ure', 'she', 'would', 'catch', 'bad', 'cold', 'if', 'she', 'did',
'not', 'get', 'dry', 'very', 'soon', 'Ahem', 'said', 'the', 'Mouse
', 'with', 'an', 'important', 'air', 'are', 'you', 'all', 'ready',
'This', 'is', 'the', 'driest', 'thing', 'know', 'Silence', 'all',
'round', 'if', 'you', 'please', 'William', 'the', 'Conqueror', 'wh
ose', 'cause', 'was', 'favoured', 'by', 'the', 'pope', 'was', 'soo
n', 'submitted', 'to', 'by', 'the', 'English', 'who', 'wanted', 'l
eaders', 'and', 'had', 'been', 'of', 'late', 'much', 'accustomed',
'to', 'usurpation', 'and', 'conquest', 'Edwin', 'and', 'Morcar', '
the', 'earls', 'of', 'Mercia', 'and', 'Northumbria', 'Ugh', 'said'
, 'the', 'Lory', 'with', 'shiver', 'I', 'beg', 'your', 'pardon', '
said', 'the', 'Mouse', 'frowning', 'but', 'very', 'politely', 'Did
', 'you', 'speak', 'Not', 'said', 'the', 'Lory', 'hastily', 'I', '
thought', 'you', 'did', 'said', 'the', 'Mouse', 'proceed', 'Edwin'
, 'and', 'Morcar', 'the', 'earls', 'of', 'Mercia', 'and', 'Northum
bria', 'declared', 'for', 'him', 'and', 'even', 'Stigand', 'the',
'patriotic', 'archbishop', 'of', 'Canterbury', 'found', 'it', 'adv
isable', 'Found', 'what', 'said', 'the', 'Duck', 'Found', 'it', 't
he', 'Mouse', 'replied', 'rather', 'crossly', 'of', 'course', 'you
', 'know', 'what', 'it', 'means', 'I', 'know', 'what', 'it', 'mean
s', 'well', 'enough', 'when', 'find', 'thing', 'said', 'the', 'Duc
k', 'it', 'generally', 'frog', 'or', 'worm', 'The', 'question', 'i
s', 'what', 'did', 'the', 'archbishop', 'find']
tokens conservés de catégorie NN ou VB
[('curiouser', 'NN'), ('moment', 'NN'), ('speak', 'VB'), ('telesco
pe', 'NN'), ('bye', 'NN'), ('be', 'VB'), ('sight', 'NN'), ('put',
'VB'), ('shan', 'NN'), ('be', 'VB'), ('be', 'VB'), ('deal', 'VB'),
('trouble', 'NN'), ('manage', 'VB'), ('way', 'NN'), ('be', 'VB'),
('kind', 'NN'), ('way', 'NN'), ('go', 'VB'), ('Let', 'VB'), ('see'
, 'VB'), ('give', 'VB'), ('pair', 'NN'), ('nAnd', 'NN'), ('plannin
g', 'NN'), ('herself', 'VB'), ('manage', 'VB'), ('go', 'VB'), ('ca
rrier', 'NN'), ('look', 'VB'), ('m', 'NN'), ('time', 'NN'), ('patt
ering', 'NN'), ('distance', 'NN'), ('see', 'VB'), ('pair', 'NN'),
('kid', 'NN'), ('hand', 'NN'), ('fan', 'NN'), ('hurry', 'NN'), ('b
e', 'VB'), ('ask', 'VB'), ('help', 'NN'), ('voice', 'NN'), ('sir',
'VB'), ('kid', 'NN'), ('fan', 'NN'), ('darkness', 'NN'), ('go', 'V
B'), ('party', 'NN'), ('bank', 'NN'), ('fur', 'NN'), ('cross', 'NN
'), ('nThe', 'NN'), ('question', 'NN'), ('course', 'NN'), ('get',
'VB'), ('consultation', 'NN'), ('find', 'VB'), ('life', 'NN'), ('a
rgument', 'NN'), ('sulky', 'NN'), ('say', 'VB'), ('know', 'VB'), (
'allow', 'VB'), ('tell', 'VB'), ('age', 'NN'), ('be', 'VB'), ('be'
```

```
, 'VB'), ('person', 'NN'), ('authority', 'NN'), ('listen', 'VB'),
('ring', 'NN'), ('catch', 'VB'), ('cold', 'NN'), ('get', 'VB'), ('
air', 'NN'), ('thing', 'NN'), ('round', 'NN'), ('cause', 'NN'), ('
pope', 'NN'), ('usurpation', 'NN'), ('conquest', 'NN'), ('earls',
'NN'), ('shiver', 'NN'), ('pardon', 'NN'), ('frowning', 'NN'), ('p
roceed', 'NN'), ('earls', 'NN'), ('archbishop', 'NN'), ('course',
'NN'), ('find', 'VB'), ('thing', 'NN'), ('worm', 'VB'), ('question
', 'NN'), ('archbishop', 'NN'), ('find', 'VB')]
Racinisation
['curious', 'moment', 'speak', 'telescop', 'bye', 'be', 'sight', '
put', 'shan', 'be', 'be', 'deal', 'troubl', 'manag', 'way', 'be',
'kind', 'way', 'go', 'let', 'see', 'give', 'pair', 'nand', 'plan',
'herself', 'manag', 'go', 'carrier', 'look', 'm', 'time', 'patter'
, 'distanc', 'see', 'pair', 'kid', 'hand', 'fan', 'hurri', 'be', '
ask', 'help', 'voic', 'sir', 'kid', 'fan', 'dark', 'go', 'parti',
'bank', 'fur', 'cross', 'nthe', 'question', 'cours', 'get', 'consu
lt', 'find', 'life', 'argument', 'sulki', 'say', 'know', 'allow',
'tell', 'age', 'be', 'be', 'person', 'author', 'listen', 'ring', '
catch', 'cold', 'get', 'air', 'thing', 'round', 'caus', 'pope', 'u
surp', 'conquest', 'earl', 'shiver', 'pardon', 'frown', 'proceed',
'earl', 'archbishop', 'cours', 'find', 'thing', 'worm', 'question'
, 'archbishop', 'find']
```

Nous savons maintenant nettoyer nos données et les transformer sous la forme de tokens. Aussi nous allons comment ces derniers peuvent être utilisés comme représentation des documents pour faire .... de la classification. La classification de données textuelles est présentée dans un autre notebook.