Démonstration NLTK

Projet tutoré FIE5 ISIS 2023

AYMARD Marion | JOLIVET Anaëlle | BOURDARIAS Cléa

Entrée [2]:
```
# Import NLTK
import nltk
from nltk import word_tokenize, pos_tag, ne_chunk, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/anaellejolivet/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[2]:
```
True
```

On utilise la phrase **'Je me suis fait mal au poignet hier après-midi.'**

Entrée [3]:
```
sentence = "Je me suis fait mal au poignet hier après-midi."
```

Prédire les types de mots dans la phrase

Entrée [4]:
```
words = word_tokenize(sentence)
pos_tags = pos_tag(words)
ne_tree = ne_chunk(pos_tags)

# Affichez les résultats
print(ne_tree)
```

```
(S
  Je/NNP
  me/PRP
  suis/JJ
  fait/NN
  mal/JJ
  au/JJ
  poignet/NN
  hier/JJR
  après-midi/NN
  ./.)
```

Prédire comment les mots sont reliés

Dans spaCy, token.head.text renvoie le texte du mot (token) tête (head) d'un token donné dans une phrase analysée. Cela permet d'accéder au texte du mot gouverneur (head) d'un mot donné dans une relation de dépendance.

En utilisant NLTK, on peut obtenir une fonctionnalité similaire en utilisant la bibliothèque de dépendance Stanford Parser, qui peut être utilisée pour analyser la structure de dépendance des phrases.

Prédire des similitudes entre les phrases

On prend les phrases **'J'ai mal aux dents.'** et **'J'ai une douleur dentaire'**. On peut determiner un taux de similitude (cosinus) entre ces phrases.
Ce taux varie entre -1 et 1.

Entrée [5]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Exemple de deux phrases
sentence1 = "J'ai mal aux dents."
sentence2 = "J'ai une douleur dentaire."

# Utilisez TF-IDF pour vectoriser les phrases
vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform([sentence1, sentence2])

# Calculez la similarité de cosinus entre les vecteurs
cosine_similarity_score = cosine_similarity(vectors[0], vectors[1])[0][0]
print("Cosine Similarity:", cosine_similarity_score)
```

```
Cosine Similarity: 0.14438355527738675
```

On prend les phrases **'J'ai mal aux dents'** et **'J'ai une douleur aux pieds'**.

Entrée [6]:
```
# Exemple de deux phrases
sentence1 = "J'ai mal aux dents."
sentence2 = "J'ai une douleur aux pieds."

# Utilisez TF-IDF pour vectoriser les phrases
vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform([sentence1, sentence2])

# Calculez la similarité de cosinus entre les vecteurs
cosine_similarity_score = cosine_similarity(vectors[0], vectors[1])[0][0]
print("Cosine Similarity:", cosine_similarity_score)
```

```
Cosine Similarity: 0.29121941856368966
```

On prend les phrases **'Je n'ai pas mal aux dents'** et **'J'ai mal aux dents'**.

Entrée [7]:
```
# Exemple de deux phrases
sentence1 = "J'ai mal aux dents."
sentence2 = "Je n'ai pas mal aux dents."

# Utilisez TF-IDF pour vectoriser les phrases
vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform([sentence1, sentence2])

# Calculez la similarité de cosinus entre les vecteurs
cosine_similarity_score = cosine_similarity(vectors[0], vectors[1])[0][0]
print("Cosine Similarity:", cosine_similarity_score)
```

```
Cosine Similarity: 0.7092972666062738
```

Comme pour Spacy, cette fonction n'est pas forcement pertinante car la similarité se fait essentiellement sur les mots et lettres communes et non pas sur le sens.

Texte Doctissimo

Entrée [8]:
```
texte = "Depuis environs 3 ans j'ai une énorme douleurs au sternum qui m'irradie dans le dos mais qui se
print(texte)
```

```
Depuis environs 3 ans j'ai une énorme douleurs au sternum qui m'irradie dans le dos mais qui se calme v
oir disparaît le plus souvent en buvant de l'eau. Depuis 1 semaines la douleurs revient plus souvent et
me réveil même la nuit je suis donc obligé de boire pour me soulager. Et aujourd'hui en buvant de l'eau
car je sentais que ça arrivé j'avais comme une sensation de boule qui se débloque entre l'oesophage et
l'estomac. Cela pourrait être seulement un symptôme d'un reflux ? Pour info il y a 1 an j'ai consulté u
n gastro qui m'a fait une gastroscopie. Très mauvais souvenir sans anesthesie car je me débattais dans
tout les sens. Le gastro à apparemment seulement vu une béance cardiaque et m'a dit de perdre du poid.
J'ai 35ans 1m94 105kg
```

Tokenizing

Entrée [9]:
```
# Sépare les phrases dans un texte
sent_tokenize(texte)
```

Out[9]:
```
["Depuis environs 3 ans j'ai une énorme douleurs au sternum qui m'irradie dans le dos mais qui se calme
voir disparaît le plus souvent en buvant de l'eau.",
 'Depuis 1 semaines la douleurs revient plus souvent et me réveil même la nuit je suis donc obligé de b
oire pour me soulager.',
 "Et aujourd'hui en buvant de l'eau car je sentais que ça arrivé j'avais comme une sensation de boule q
ui se débloque entre l'oesophage et l'estomac.",
 "Cela pourrait être seulement un symptôme d'un reflux ?",
 "Pour info il y a 1 an j'ai consulté un gastro qui m'a fait une gastroscopie.",
 'Très mauvais souvenir sans anesthesie car je me débattais dans tout les sens.',
 "Le gastro à apparemment seulement vu une béance cardiaque et m'a dit de perdre du poid.",
 "J'ai 35ans 1m94 105kg"]
```

Entrée [10]:
```
# Sépare les mots
texte_tokenize = word_tokenize(texte)
print(len(texte_tokenize))
```

```
144
```

Filtering Stop Words

Entrée [11]:
```
# Obtenez la liste des mots vides en français
french_stopwords = set(stopwords.words('french'))
```

Entrée [12]:
```
filtered_list = []
for word in texte_tokenize:
    if word.casefold() not in french_stopwords:
        filtered_list.append(word)
print(filtered_list)
print(len(filtered_list))
```

```
['Depuis', 'environs', '3', 'ans', "j'ai", 'énorme', 'douleurs', 'sternum', "m'irradie", 'dos', 'calm
e', 'voir', 'disparaît', 'plus', 'souvent', 'buvant', "l'eau", '.', 'Depuis', '1', 'semaines', 'douleur
s', 'revient', 'plus', 'souvent', 'réveil', 'nuit', 'donc', 'obligé', 'boire', 'soulager', '.', "aujour
d'hui", 'buvant', "l'eau", 'car', 'sentais', 'ça', 'arrivé', "j'avais", 'comme', 'sensation', 'boule',
'débloque', 'entre', "l'oesophage", "l'estomac", '.', 'Cela', 'pourrait', 'être', 'seulement', 'symptôm
e', "d'un", 'reflux', '?', 'info', 'a', '1', 'an', "j'ai", 'consulté', 'gastro', "'", 'a', 'fait', 'gas
troscopie', '.', 'Très', 'mauvais', 'souvenir', 'sans', 'anesthesie', 'car', 'débattais', 'tout', 'sen
s', '.', 'gastro', 'apparemment', 'seulement', 'vu', 'béance', 'cardiaque', "'", 'a', 'dit', 'perdre',
'poid', '.', "J'ai", '35ans', '1m94', '105kg']
94
```

Entrée [13]:
```
filtered_words = [word for word in texte_tokenize if word not in french_stopwords]
print(filtered_words)
print(len(filtered_words))
```

```
['Depuis', 'environs', '3', 'ans', "j'ai", 'énorme', 'douleurs', 'sternum', "m'irradie", 'dos', 'calm
e', 'voir', 'disparaît', 'plus', 'souvent', 'buvant', "l'eau", '.', 'Depuis', '1', 'semaines', 'douleur
s', 'revient', 'plus', 'souvent', 'réveil', 'nuit', 'donc', 'obligé', 'boire', 'soulager', '.', 'Et',
"aujourd'hui", 'buvant', "l'eau", 'car', 'sentais', 'ça', 'arrivé', "j'avais", 'comme', 'sensation', 'b
oule', 'débloque', 'entre', "l'oesophage", "l'estomac", '.', 'Cela', 'pourrait', 'être', 'seulement',
'symptôme', "d'un", 'reflux', '?', 'Pour', 'info', 'a', '1', 'an', "j'ai", 'consulté', 'gastro', "'",
'a', 'fait', 'gastroscopie', '.', 'Très', 'mauvais', 'souvenir', 'sans', 'anesthesie', 'car', 'débattai
s', 'tout', 'sens', '.', 'Le', 'gastro', 'apparemment', 'seulement', 'vu', 'béance', 'cardiaque', "'",
'a', 'dit', 'perdre', 'poid', '.', "J'ai", '35ans', '1m94', '105kg']
97
```

Entrée [14]:
```
différence = [word for word in filtered_words if word not in filtered_list]
print(différence)
```

```
['Et', 'Pour', 'Le']
```

Stemming

Entrée [15]:
```
# Réduit un mot à sa racine
stemmer = PorterStemmer()
```

Entrée [16]:
```
phrase1 = "Depuis environs 3 ans j'ai une énorme douleurs au sternum qui m'irradie dans le dos mais qui
phrase1_tokenize = word_tokenize(phrase1)
stemmed_phrase1 = [stemmer.stem(word) for word in phrase1_tokenize]
print(phrase1_tokenize)
print(stemmed_phrase1)
```

```
['Depuis', 'environs', '3', 'ans', "j'ai", 'une', 'énorme', 'douleurs', 'au', 'sternum', 'qui', "m'irra
die", 'dans', 'le', 'dos', 'mais', 'qui', 'se', 'calme', 'voir', 'disparaît', 'le', 'plus', 'souvent',
'en', 'buvant', 'de', "l'eau", '.']
['depui', 'environ', '3', 'an', "j'ai", 'une', 'énorm', 'douleur', 'au', 'sternum', 'qui', "m'irradi",
'dan', 'le', 'do', 'mai', 'qui', 'se', 'calm', 'voir', 'disparaît', 'le', 'plu', 'souvent', 'en', 'buva
nt', 'de', "l'eau", '.']
```

Tagging Parts of Speech (POS)

Entrée [17]:
```
# Analyse grammaticale
nltk.download('averaged_perceptron_tagger')
POS_phrase1 = nltk.pos_tag(phrase1_tokenize)
print(POS_phrase1)
```

```
[('Depuis', 'NNP'), ('environs', 'NNS'), ('3', 'CD'), ('ans', 'NNS'), ("j'ai", 'JJ'), ('une', 'JJ'),
('énorme', 'NN'), ('douleurs', 'NNS'), ('au', 'VBP'), ('sternum', 'NN'), ('qui', 'NN'), ("m'irradie",
'NN'), ('dans', 'NNS'), ('le', 'VBP'), ('dos', 'JJ'), ('mais', 'NN'), ('qui', 'NN'), ('se', 'NN'), ('ca
lme', 'NN'), ('voir', 'NN'), ('disparaît', 'NN'), ('le', 'NN'), ('plus', 'CC'), ('souvent', 'JJ'), ('e
n', 'NN'), ('buvant', 'NN'), ('de', 'IN'), ("l'eau", 'NN'), ('.', '.')]
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/anaellejolivet/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Lemmatizing

Entrée [18]:
```
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

# Obtenir la racine d'un mot (qui a du sens)
lemmatizer = WordNetLemmatizer()


# Fonction pour mapper les étiquettes POS personnalisées aux étiquettes POS de WordNet
def get_wordnet_pos_custom(custom_pos):
    if custom_pos == "a":
        return wordnet.ADJ
    elif custom_pos == "n":
        return wordnet.NOUN
    elif custom_pos == "v":
        return wordnet.VERB
    elif custom_pos == "r":
        return wordnet.ADV
    else:
        return wordnet.NOUN  # Par défaut, traitons les mots comme des noms


lemmatized_phrase1 = [lemmatizer.lemmatize(word, get_wordnet_pos_custom(pos_tag)) for word, pos_tag in

print(lemmatized_phrase1)
print("Elements changés", set(phrase1_tokenize) - set(lemmatized_phrase1))
```

```
['Depuis', 'environs', '3', 'an', "j'ai", 'une', 'énorme', 'douleurs', 'au', 'sternum', 'qui', "m'irrad
ie", 'dans', 'le', 'do', 'mais', 'qui', 'se', 'calme', 'voir', 'disparaît', 'le', 'plus', 'souvent', 'e
n', 'buvant', 'de', "l'eau", '.']
Elements changés {'ans', 'dos'}
```

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

# Téléchargez les ressources nécessaires si ce n'est pas déjà fait
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')

# Exemple de phrase
phrase = "Les chats noirs étaient en train de courir rapidement dans la rue."

# Tokenisation des mots
```

```
phrase_tokenize = word_tokenize(phrase, language='french')  # Spécifiez la langue pour la tokenisation
en français

# Étiquetage des parties du discours avec NLTK
pos_tags = nltk.pos_tag(phrase_tokenize, lang='fra')  # Spécifiez la langue pour l'étiquetage en
français

# Initialisez le lemmatiseur WordNet
lemmatizer = WordNetLemmatizer()

# Fonction pour mapper les étiquettes POS de NLTK aux étiquettes POS de WordNet en français
def get_wordnet_pos(nltk_tag):
    if nltk_tag.startswith('A'):
        return 'a'  # Adjectif
    elif nltk_tag.startswith('N'):
        return 'n'  # Nom
    elif nltk_tag.startswith('V'):
        return 'v'  # Verbe
    elif nltk_tag.startswith('R'):
        return 'r'  # Adverbe
    else:
        return None

# Lemmatisez les mots en utilisant les étiquettes POS
lemmatized_phrase = []
for word, pos_tag in pos_tags:
    wordnet_pos = get_wordnet_pos(pos_tag)
    if wordnet_pos:
        lemmatized_word = lemmatizer.lemmatize(word, wordnet_pos)
    else:
        lemmatized_word = word
    lemmatized_phrase.append(lemmatized_word)

# Reconstituez la phrase lemmatisée
lemmatized_text = ' '.join(lemmatized_phrase)

print("Phrase lemmatisée:", lemmatized_text)
```

Traduction deepl

Entrée [32]:
```
texte_anglais = "For about 3 years I've had a huge pain in my sternum that radiates down my back but ca
print(texte_anglais)
```

```
For about 3 years I've had a huge pain in my sternum that radiates down my back but calms down and even
disappears most of the time when I drink water. For the past 1 week the pain has been coming back more
often and waking me up even at night, so I've had to drink to relieve it. And today, when I was drinkin
g water because I felt it coming on, I had a sensation of a ball unblocking between the esophagus and t
he stomach. Could this just be a symptom of reflux? For your information, 1 year ago I consulted a gast
ro who did a gastroscopy. Very bad memory without anesthesia because I was struggling in all direction
s. The gastro apparently only saw a cardiac gap and told me to lose weight. I'm 35 years old, 1m94, 105
kg.
```

Tokenizing

Entrée [33]:
```
# Sépare les phrases dans un texte
sent_tokenize(texte_anglais)
```

Out[33]:
```
["For about 3 years I've had a huge pain in my sternum that radiates down my back but calms down and ev
en disappears most of the time when I drink water.",
 "For the past 1 week the pain has been coming back more often and waking me up even at night, so I've
had to drink to relieve it.",
 'And today, when I was drinking water because I felt it coming on, I had a sensation of a ball unblock
ing between the esophagus and the stomach.',
 'Could this just be a symptom of reflux?',
 'For your information, 1 year ago I consulted a gastro who did a gastroscopy.',
 'Very bad memory without anesthesia because I was struggling in all directions.',
 'The gastro apparently only saw a cardiac gap and told me to lose weight.',
 "I'm 35 years old, 1m94, 105kg."]
```

Entrée [34]:
```
# Sépare les mots
texte_tokenize_anglais = word_tokenize(texte_anglais)
```

Filtering Stop Words

Entrée [35]:
```
stop_words = set(stopwords.words('english'))

filtered_sentence = [w for w in texte_tokenize_anglais if not w.lower() in stop_words]
#with no lower case conversion
filtered_sentence = []

for w in texte_tokenize_anglais:
    if w not in stop_words:
        filtered_sentence.append(w)

print(len(texte_tokenize_anglais))
print(len(filtered_sentence))
print(filtered_sentence)
```

```
157
96
['For', '3', 'years', 'I', "'ve", 'huge', 'pain', 'sternum', 'radiates', 'back', 'calms', 'even', 'disa
ppears', 'time', 'I', 'drink', 'water', '.', 'For', 'past', '1', 'week', 'pain', 'coming', 'back', 'oft
en', 'waking', 'even', 'night', ',', 'I', "'ve", 'drink', 'relieve', '.', 'And', 'today', ',', 'I', 'dr
inking', 'water', 'I', 'felt', 'coming', ',', 'I', 'sensation', 'ball', 'unblocking', 'esophagus', 'sto
mach', '.', 'Could', 'symptom', 'reflux', '?', 'For', 'information', ',', '1', 'year', 'ago', 'I', 'con
sulted', 'gastro', 'gastroscopy', '.', 'Very', 'bad', 'memory', 'without', 'anesthesia', 'I', 'struggli
ng', 'directions', '.', 'The', 'gastro', 'apparently', 'saw', 'cardiac', 'gap', 'told', 'lose', 'weigh
t', '.', 'I', "'m", '35', 'years', 'old', ',', '1m94', ',', '105kg', '.']
```

Stemming

Entrée [36]:
```
# Réduit un mot à sa racine
stemmer = PorterStemmer()

stemmed_texte_anglais = [stemmer.stem(word) for word in texte_tokenize_anglais]
print(texte_tokenize_anglais)
print(stemmed_texte_anglais)
```

```
['For', 'about', '3', 'years', 'I', "'ve", 'had', 'a', 'huge', 'pain', 'in', 'my', 'sternum', 'that',
'radiates', 'down', 'my', 'back', 'but', 'calms', 'down', 'and', 'even', 'disappears', 'most', 'of', 't
he', 'time', 'when', 'I', 'drink', 'water', '.', 'For', 'the', 'past', '1', 'week', 'the', 'pain', 'ha
s', 'been', 'coming', 'back', 'more', 'often', 'and', 'waking', 'me', 'up', 'even', 'at', 'night', ',',
'so', 'I', "'ve", 'had', 'to', 'drink', 'to', 'relieve', 'it', '.', 'And', 'today', ',', 'when', 'I',
'was', 'drinking', 'water', 'because', 'I', 'felt', 'it', 'coming', 'on', ',', 'I', 'had', 'a', 'sensat
ion', 'of', 'a', 'ball', 'unblocking', 'between', 'the', 'esophagus', 'and', 'the', 'stomach', '.', 'Co
uld', 'this', 'just', 'be', 'a', 'symptom', 'of', 'reflux', '?', 'For', 'your', 'information', ',',
'1', 'year', 'ago', 'I', 'consulted', 'a', 'gastro', 'who', 'did', 'a', 'gastroscopy', '.', 'Very', 'ba
d', 'memory', 'without', 'anesthesia', 'because', 'I', 'was', 'struggling', 'in', 'all', 'directions',
'.', 'The', 'gastro', 'apparently', 'only', 'saw', 'a', 'cardiac', 'gap', 'and', 'told', 'me', 'to', 'l
ose', 'weight', '.', 'I', "'m", '35', 'years', 'old', ',', '1m94', ',', '105kg', '.']
['for', 'about', '3', 'year', 'i', "'ve", 'had', 'a', 'huge', 'pain', 'in', 'my', 'sternum', 'that', 'r
adiat', 'down', 'my', 'back', 'but', 'calm', 'down', 'and', 'even', 'disappear', 'most', 'of', 'the',
'time', 'when', 'i', 'drink', 'water', '.', 'for', 'the', 'past', '1', 'week', 'the', 'pain', 'ha', 'be
en', 'come', 'back', 'more', 'often', 'and', 'wake', 'me', 'up', 'even', 'at', 'night', ',', 'so', 'i',
"'ve", 'had', 'to', 'drink', 'to', 'reliev', 'it', '.', 'and', 'today', ',', 'when', 'i', 'wa', 'drin
k', 'water', 'becaus', 'i', 'felt', 'it', 'come', 'on', ',', 'i', 'had', 'a', 'sensat', 'of', 'a', 'bal
l', 'unblock', 'between', 'the', 'esophagu', 'and', 'the', 'stomach', '.', 'could', 'thi', 'just', 'b
e', 'a', 'symptom', 'of', 'reflux', '?', 'for', 'your', 'inform', ',', '1', 'year', 'ago', 'i', 'consul
t', 'a', 'gastro', 'who', 'did', 'a', 'gastroscopi', '.', 'veri', 'bad', 'memori', 'without', 'anesthes
ia', 'becaus', 'i', 'wa', 'struggl', 'in', 'all', 'direct', '.', 'the', 'gastro', 'appar', 'onli', 'sa
w', 'a', 'cardiac', 'gap', 'and', 'told', 'me', 'to', 'lose', 'weight', '.', 'i', "'m", '35', 'year',
'old', ',', '1m94', ',', '105kg', '.']
```

Tagging Parts of Speech (POS)

Entrée [37]:
```
# Analyse grammaticale
POS_texte_anglais = nltk.pos_tag(texte_tokenize_anglais)
print(POS_texte_anglais)
```

```
[('For', 'IN'), ('about', 'RB'), ('3', 'CD'), ('years', 'NNS'), ('I', 'PRP'), ("'ve", 'VBP'), ('had',
'VBD'), ('a', 'DT'), ('huge', 'JJ'), ('pain', 'NN'), ('in', 'IN'), ('my', 'PRP$'), ('sternum', 'NN'),
('that', 'IN'), ('radiates', 'VBZ'), ('down', 'RP'), ('my', 'PRP$'), ('back', 'NN'), ('but', 'CC'), ('c
alms', 'VBP'), ('down', 'RP'), ('and', 'CC'), ('even', 'RB'), ('disappears', 'VBZ'), ('most', 'JJS'),
('of', 'IN'), ('the', 'DT'), ('time', 'NN'), ('when', 'WRB'), ('I', 'PRP'), ('drink', 'VBP'), ('water',
'NN'), ('.', '.'), ('For', 'IN'), ('the', 'DT'), ('past', 'JJ'), ('1', 'CD'), ('week', 'NN'), ('the',
'DT'), ('pain', 'NN'), ('has', 'VBZ'), ('been', 'VBN'), ('coming', 'VBG'), ('back', 'RB'), ('more', 'RB
R'), ('often', 'RB'), ('and', 'CC'), ('waking', 'VBG'), ('me', 'PRP'), ('up', 'IN'), ('even', 'RB'),
('at', 'IN'), ('night', 'NN'), (',', ','), ('so', 'IN'), ('I', 'PRP'), ("'ve", 'VBP'), ('had', 'VBN'),
('to', 'TO'), ('drink', 'VB'), ('to', 'TO'), ('relieve', 'VB'), ('it', 'PRP'), ('.', '.'), ('And', 'C
C'), ('today', 'NN'), (',', ','), ('when', 'WRB'), ('I', 'PRP'), ('was', 'VBD'), ('drinking', 'VBG'),
('water', 'NN'), ('because', 'IN'), ('I', 'PRP'), ('felt', 'VBD'), ('it', 'PRP'), ('coming', 'VBG'),
('on', 'IN'), (',', ','), ('I', 'PRP'), ('had', 'VBD'), ('a', 'DT'), ('sensation', 'NN'), ('of', 'IN'),
('a', 'DT'), ('ball', 'NN'), ('unblocking', 'NN'), ('between', 'IN'), ('the', 'DT'), ('esophagus', 'N
N'), ('and', 'CC'), ('the', 'DT'), ('stomach', 'NN'), ('.', '.'), ('Could', 'VB'), ('this', 'DT'), ('ju
st', 'RB'), ('be', 'VB'), ('a', 'DT'), ('symptom', 'NN'), ('of', 'IN'), ('reflux', 'NN'), ('?', '.'),
('For', 'IN'), ('your', 'PRP$'), ('information', 'NN'), (',', ','), ('1', 'CD'), ('year', 'NN'), ('ag
o', 'RB'), ('I', 'PRP'), ('consulted', 'VBD'), ('a', 'DT'), ('gastro', 'NN'), ('who', 'WP'), ('did', 'V
BD'), ('a', 'DT'), ('gastroscopy', 'NN'), ('.', '.'), ('Very', 'RB'), ('bad', 'JJ'), ('memory', 'NN'),
('without', 'IN'), ('anesthesia', 'NN'), ('because', 'IN'), ('I', 'PRP'), ('was', 'VBD'), ('strugglin
g', 'VBG'), ('in', 'IN'), ('all', 'DT'), ('directions', 'NNS'), ('.', '.'), ('The', 'DT'), ('gastro',
'NN'), ('apparently', 'RB'), ('only', 'RB'), ('saw', 'VBD'), ('a', 'DT'), ('cardiac', 'JJ'), ('gap', 'N
N'), ('and', 'CC'), ('told', 'VBD'), ('me', 'PRP'), ('to', 'TO'), ('lose', 'VB'), ('weight', 'NN'),
('.', '.'), ('I', 'PRP'), ("'m", 'VBP'), ('35', 'CD'), ('years', 'NNS'), ('old', 'JJ'), (',', ','), ('1
m94', 'CD'), (',', ','), ('105kg', 'CD'), ('.', '.')]
```

Lemmatizing

Entrée [38]:
```
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

# Obtenir la racine d'un mot (qui a du sens)
lemmatizer = WordNetLemmatizer()


# Fonction pour mapper les étiquettes POS personnalisées aux étiquettes POS de WordNet
def get_wordnet_pos_custom(custom_pos):
    if custom_pos == "a":
        return wordnet.ADJ
    elif custom_pos == "n":
        return wordnet.NOUN
    elif custom_pos == "v":
        return wordnet.VERB
    elif custom_pos == "r":
        return wordnet.ADV
    else:
        return wordnet.NOUN  # Par défaut, traitons les mots comme des noms


lemmatized_texte_anglais = [lemmatizer.lemmatize(word, get_wordnet_pos_custom(pos_tag)) for word, pos_t

print(lemmatized_texte_anglais)
print("Elements changés", set(texte_tokenize_anglais) - set(lemmatized_texte_anglais))
```

```
['For', 'about', '3', 'year', 'I', "'ve", 'had', 'a', 'huge', 'pain', 'in', 'my', 'sternum', 'that', 'r
adiates', 'down', 'my', 'back', 'but', 'calm', 'down', 'and', 'even', 'disappears', 'most', 'of', 'th
e', 'time', 'when', 'I', 'drink', 'water', '.', 'For', 'the', 'past', '1', 'week', 'the', 'pain', 'ha',
'been', 'coming', 'back', 'more', 'often', 'and', 'waking', 'me', 'up', 'even', 'at', 'night', ',', 's
o', 'I', "'ve", 'had', 'to', 'drink', 'to', 'relieve', 'it', '.', 'And', 'today', ',', 'when', 'I', 'w
a', 'drinking', 'water', 'because', 'I', 'felt', 'it', 'coming', 'on', ',', 'I', 'had', 'a', 'sensatio
n', 'of', 'a', 'ball', 'unblocking', 'between', 'the', 'esophagus', 'and', 'the', 'stomach', '.', 'Coul
d', 'this', 'just', 'be', 'a', 'symptom', 'of', 'reflux', '?', 'For', 'your', 'information', ',', '1',
'year', 'ago', 'I', 'consulted', 'a', 'gastro', 'who', 'did', 'a', 'gastroscopy', '.', 'Very', 'bad',
'memory', 'without', 'anesthesia', 'because', 'I', 'wa', 'struggling', 'in', 'all', 'direction', '.',
'The', 'gastro', 'apparently', 'only', 'saw', 'a', 'cardiac', 'gap', 'and', 'told', 'me', 'to', 'lose',
'weight', '.', 'I', "'m", '35', 'year', 'old', ',', '1m94', ',', '105kg', '.']
Elements changés {'calms', 'was', 'has', 'directions', 'years'}
```

Parse tree

Entrée [43]:
```
from nltk.corpus import treebank
words = word_tokenize(sentence)
tagged = pos_tag(words)
named_entities = ne_chunk(tagged)

# Créez un arbre d'analyse syntaxique à partir de l'analyse en entités nommées
sentence_tree = tree.Tree.fromstring(named_entities.pformat())

# Affichez l'arbre d'analyse syntaxique
print(sentence_tree)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/w_/bqmq_6m1555gjd4cljgf7ppc0000gn/T/ipykernel_82712/2419913834.py in <module>
      5
      6 # Créez un arbre d'analyse syntaxique à partir de l'analyse en entités nommées
----> 7 sentence_tree = tree.Tree.fromstring(named_entities.pformat())
      8
      9 # Affichez l'arbre d'analyse syntaxique

NameError: name 'tree' is not defined
```

Entrée [ ]: