Atelier 1 : Techniques NLP de Base

1. Objectif

L'objectif de cet atelier est d'apprendre les tâches NLP les plus courantes à travers l'utilisation des bibliothèques nltk, scikitlearn et Spacy.

2. Outils et environnement de travail

Installer les package nltk.

```
import nltk
# nltk.download('all')

#nltk.download('punkt')
```

3. Segmentation (Tokenization)

La segmentation de texte et la tâche de subdivision du texte en petites unités qui seront plus simples à traiter et qu'on appelle tokens. La bibliothèque nltk offre à travers le module **tekenize** un certain nombre de tokinzers qui permettent de réaliser la segmentation du texte en fonction de la nature du problème : words tokenizer, regular-expression based tokenizer, sentences based tokinizers, etc. Ci-dessous une liste non exhaustive de quelques fonctions du module tokinize.

- regexp_span_tokenize(text, regexp): Retourne les tokens de texte qui correspondent à l'expression régulière regexp
- sent_tokenize(text[, language]): Retourne les phrases contenues dans le texte en utilisant le tokenizer PunktSentenceTokenizer.
- word_tokenize(text[, language]: Retourne les mots contenus dans le texte en utilisant le tokenizer TreebankWordTokenizer avec PunktSentenceTokenizer.

NLTK offre également un certain nombre de classes qui offrent des tokinizers plus avancés : BlanklineTokenizer, MWETokenizer, PunktSentenceTokenizer, TextTilingTokenizer, TweetTokenizer, etc. Ci-dessous deux exemples de tokenization à base de **sent_tokenize** et **word_tokenize**:

```
from nltk.tokenize import sent_tokenize, word_tokenize
```

```
data="Hello, i am very happy to meet you. I created this course for
you. Good by!"

sentences=sent_tokenize(data)
print("sentences: " , sentences)

words=word_tokenize(data)
print("Words: ",words)
```

```
sentences:  ['Hello, i am very happy to meet you.', 'I created this
course for you.', 'Good by!']
Words:  ['Hello', ',', 'i', 'am', 'very', 'happy', 'to', 'meet',
'you', '.', 'I', 'created', 'this', 'course', 'for', 'you', '.',
'Good', 'by', '!']
```

4. Nettoyage

Le nettoyage des données texte joue un rôle très important dans l'amélioration des performances des opérations d'analyse et de découverte de paternes. Ça consiste à la suppression des termes non significatifs **"Stop words"**, comme par exemple « le », « la », « de », « du », « ce »… en français et « as » « the », « a », « an », « in » en anglais. Ces termes qui sont présents fréquemment dans des documents texte peuvent influencer négativement sur la qualité des résultats d'analyse. Le nettoyage peut consister également à la supression des caratères de pontuation et des chaînes de caractères non alphabétiques. Ci-dessous le code qui permet de supprimer les stop words à partir d'un texte.

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

data="Hello, i am very happy to meet you. I created this course for
you. Good by!"
word_tokens = [word.lower() for word in word_tokenize(data)]
data_clean = [word for word in word_tokens if (not word in
set(stopwords.words('english')) and  word.isalpha())]
print(data_clean)
```

```
['hello', 'happy', 'meet', 'created', 'course', 'good']
```

5. Racinisation

La racinisation (Stemming en anglais) permet de normaliser la représentation des mots contenus dans une expression texte en extrayant leurs racines. Ça permettra de supprimer toutes les redondances des mots ayant la même racine. Plusieurs **stemmers** sont offerts par nltk dont les plus utilisés sont : *PorterStemmer, LancasterStemmer, SnowballStemmer….* Également, le module nltk.stem.snowball offre un certain nombre de stemmers personnalisés à chaque langue, comme par exemple : *FrenchStemmer, ArabicStemmer*, etc.

```
from nltk.stem import PorterStemmer,SnowballStemmer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
#stemmer=SnowballStemmer('french')
stemmer=PorterStemmer()
data="Hello, i am very happy to meet you. I created this course for
you. Good by!"

word_tokens = [word.lower() for word in word_tokenize(data)]

for i in range(len(word_tokens)):
    words=[stemmer.stem(word) for word in word_tokens if (not word in
set(stopwords.words('english')) and  word.isalpha())]
print(words)
```

```
['hello', 'happi', 'meet', 'creat', 'cours', 'good']
```

6. Lemmatisation

A la différence de la racisation qui fournit souvent une représentation non significative et incomplète des mots, la lemmatisation permet d'obtenir les **formes canoniques** des mots contenus dans une expression texte. Ainsi, au lieu de supprimer juste les suffixes et les préfixes des mots pour obtenir leurs racines, la lemmatisation réalise une **analyse morphologique** des mots afin d'extraire leurs formats canoniques. nltk offre le lemmatizer *WordNetLemmatizer* pour la réalisation des opérations de lemmatisation, mais uniquement pour l'anglais. pour d'autre langues on peut recourir à la bibliotheque **SpaCy**.

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

lemmmatizer=WordNetLemmatizer()
data1="Le big data  « grosses données » en anglais,les mégadonnées ou
les données massives, " \
     "désigne les ressources d'informations dont les caractéristiques
en termes de volume," \
     " de vélocité et de variété imposent l'utilisation de
technologies et de méthodes analytiques " \
     "particulières pour générer de la valeur, et qui dépassent en
général les capacités " \
     "d'une seule et unique machine et nécessitent des traitements
parallélisés"
data2="Big data is a field that treats ways to analyze, systematically
extract information from," \
     " or otherwise deal with data sets that are too large or complex
to be dealt with by traditional" \
     " data-processing application software. Data with many cases
```

```
(rows) offer greater statistical power," \
     " while data with higher complexity (more attributes or columns)
" \
     "may lead to a higher false discovery rate. "

words1 = word_tokenize(data1)
words1 = [lemmmatizer.lemmatize(word.lower()) for word in words1
if(not word in set(stopwords.words('french')) and  word.isalpha())]
print(words1)

words2 = word_tokenize(data2)
words2 = [lemmmatizer.lemmatize(word.lower()) for word in words2
if(not word in set(stopwords.words('english')) and  word.isalpha())]
print(words2)
```

```
['le', 'big', 'data', 'gross', 'données', 'anglais', 'mégadonnées',
'données', 'massif', 'désigne', 'ressources', 'information', 'dont',
'caractéristiques', 'termes', 'volume', 'vélocité', 'variété',
'imposent', 'utilisation', 'technology', 'méthodes', 'analytiques',
'particulières', 'générer', 'valeur', 'dépassent', 'général',
'capacités', 'seule', 'unique', 'machine', 'nécessitent',
'traitements', 'parallélisés']
['big', 'data', 'field', 'treat', 'way', 'analyze', 'systematically',
'extract', 'information', 'otherwise', 'deal', 'data', 'set', 'large',
'complex', 'dealt', 'traditional', 'application', 'software', 'data',
'many', 'case', 'row', 'offer', 'greater', 'statistical', 'power',
'data', 'higher', 'complexity', 'attribute', 'column', 'may', 'lead',
'higher', 'false', 'discovery', 'rate']
```

#7. POS-Tagging

Le pos-tagging permet de réaliser une analyse lexicale d'une expression texte selon les règles de la grammaire. Les différentes unités seront dotées d'une annotation permettant de savoir le rôle grammatical de chaque mot dans l'expression. Les annotations les plus courante sont (DT : Determiner, NN : noun , JJ : adjective, RB: adverb, VB : verb, PRP : Personal Pronoun…).

NLTK offre une panoplie de taggers pour le pos-taggin qui recoivent une liste de tokens et leurs attribuent automatiquement des tags en se basant sur des corpus d'apprentisgae.

Par defaut la methode *pos_tag* offre un pos_tagging standard (Recommandé) pour l'anglais et cela en se bsant sur le tagset *"Penn Treebank"*:

```
import nltk
from nltk.tokenize import word_tokenize
data="Hello, i am very happy to meet you. I created this course for
you. Good by!. "
words=word_tokenize(data)
print(nltk.pos_tag(words))
```

```
[('Hello', 'NNP'), (',', ','), ('i', 'NN'), ('am', 'VBP'), ('very',
'RB'), ('happy', 'JJ'), ('to', 'TO'), ('meet', 'VB'), ('you', 'PRP'),
('.', '.'), ('I', 'PRP'), ('created', 'VBD'), ('this', 'DT'),
('course', 'NN'), ('for', 'IN'), ('you', 'PRP'), ('.', '.'), ('Good',
'JJ'), ('by', 'IN'), ('!', '.'), ('.', '.')]
```

Dans le cas d'un document qui se compose de plusieurs phrases, il sera preferable d'utliser pos_tag_sents.

```
import nltk
from nltk.tokenize import sent_tokenize
data="Hello, i am very happy to meet you. I created this course for
you. Good by!. "


sentences=sent_tokenize(data)

list=[]
for sentence in sentences:
    list.append(word_tokenize(sentence))

print(nltk.pos_tag_sents(list))
```

```
[[('Hello', 'NNP'), (',', ','), ('i', 'NN'), ('am', 'VBP'), ('very',
'RB'), ('happy', 'JJ'), ('to', 'TO'), ('meet', 'VB'), ('you', 'PRP'),
('.', '.')], [('I', 'PRP'), ('created', 'VBD'), ('this', 'DT'),
('course', 'NN'), ('for', 'IN'), ('you', 'PRP'), ('.', '.')],
[('Good', 'JJ'), ('by', 'IN'), ('!', '.'), ('.', '.')]]
```

UnigramTagger permet d'attribuer aux mots leurs tags les plus frequents par rapport à un corpus d'apprentissage.

```
import nltk
nltk.download('brown')
from nltk.corpus import brown
from nltk.tag import UnigramTagger
from nltk.tokenize import word_tokenize

brown_tagged_sents = brown.tagged_sents(categories='news')
size = int(len(brown_tagged_sents) * 0.9)
train_sents = brown_tagged_sents[:size]
test_sents = brown_tagged_sents[size:]
unigram_tagger = nltk.UnigramTagger(train_sents)
print(unigram_tagger.evaluate(test_sents))

data="Hello, i am very happy to meet you. I created this course for
you. Good by!. "
print(unigram_tagger.tag(word_tokenize(data)))
```

```
[nltk_data] Downloading package brown to
[nltk_data]     C:\Users\dscon\AppData\Roaming\nltk_data...
[nltk_data]   Package brown is already up-to-date!

0.8121200039868434
[('Hello', None), (',', ','), ('i', None), ('am', 'BEM'), ('very',
'QL'), ('happy', 'JJ'), ('to', 'TO'), ('meet', 'VB'), ('you', 'PPSS'),
('.', '.'), ('I', 'PPSS'), ('created', 'VBN'), ('this', 'DT'),
('course', 'NN'), ('for', 'IN'), ('you', 'PPSS'), ('.', '.'), ('Good',
'JJ-TL'), ('by', 'IN'), ('!', '.'), ('.', '.')]

C:\Users\dscon\AppData\Local\Temp\ipykernel_9760\829503320.py:12:
DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(unigram_tagger.evaluate(test_sents))
```

le modèle n-gram est une generalisation de l'unigram qui cnsidère également le contexte où apparait le mot en considerant les tags des n-1 mots precedents.

bigram tagger est un exemple generateur pos-tagging n-gram.

```
brown_tagged_sents = brown.tagged_sents()

size = int(len(brown_tagged_sents) * 0.9)
train_sents = brown_tagged_sents[:size]
test_sents = brown_tagged_sents[size:]

bigram_tagger = nltk.BigramTagger(train_sents)

print(bigram_tagger.evaluate(test_sents))

data="Hello, i am very happy to meet you. I created this course for
you. Good by!"
print(bigram_tagger.tag(word_tokenize(data)))

C:\Users\dscon\AppData\Local\Temp\ipykernel_9760\88343320.py:9:
DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(bigram_tagger.evaluate(test_sents))

0.3515747783994468
[('Hello', 'UH'), (',', ','), ('i', None), ('am', None), ('very',
None), ('happy', None), ('to', None), ('meet', None), ('you', None),
('.', None), ('I', None), ('created', None), ('this', None),
('course', None), ('for', None), ('you', None), ('.', None), ('Good',
None), ('by', None), ('!', None)]
```

On peut combiner plusieurs taggers en les executant d'une maniere sequentielle comme montré dans l'exemple c-dessous:

```
brown_tagged_sents = brown.tagged_sents(categories='news')

size = int(len(brown_tagged_sents) * 0.9)
train_sents = brown_tagged_sents[:size]
test_sents = brown_tagged_sents[size:]

t0 = nltk.DefaultTagger('NN')
t1 = nltk.UnigramTagger(train_sents, backoff=t0)
t2 = nltk.BigramTagger(train_sents, backoff=t1)
print(t2.evaluate(test_sents))

data="Hello, i am very happy to meet you. I created this course for
you. Good by!"
print(t2.tag(word_tokenize(data)))
```

```
0.8452108043456593
[('Hello', 'NN'), (',', ','), ('i', 'NN'), ('am', 'BEM'), ('very',
'QL'), ('happy', 'JJ'), ('to', 'TO'), ('meet', 'VB'), ('you', 'PPO'),
('.', '.'), ('I', 'PPSS'), ('created', 'VBN'), ('this', 'DT'),
('course', 'NN'), ('for', 'IN'), ('you', 'PPO'), ('.', '.'), ('Good',
'JJ-TL'), ('by', 'IN'), ('!', '.')]

C:\Users\dscon\AppData\Local\Temp\ipykernel_9760\1528170503.py:10:
DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(t2.evaluate(test_sents))
```

Pour le moment la package nltk ne permet de faire le pos-tagging que pour l'anglais et le russe à l'aide du modèle « averaged_perceptron_tagger ». StanfordPOSTagguer permet faire du pos-tagging pour d'autre langues comme le français et l'arabe. Il suffit de télécharger les differents librairies nécessaires (https://nlp.stanford.edu/software/tagger.shtml) et utiliser celles qui correspondent à la langue comme présenté dans l'exemple ci-dessous.

```
# !unzip stanford-tagger-4.2.0
```

```
'unzip' n'est pas reconnu en tant que commande interne
ou externe, un programme ex□cutable ou un fichier de commandes.
```

```
import nltk
from nltk.tag.stanford import StanfordPOSTagger

data="Le big data  « grosses données » en anglais,les mégadonnées ou
les données massives, " \
     "désigne les ressources d'informations dont les caractéristiques
en termes de volume," \
     " de vélocité et de variété imposent l'utilisation de
technologies et de méthodes analytiques " \
     "particulières pour générer de la valeur, et qui dépassent en
```

```
général les capacités " \
     "d'une seule et unique machine et nécessitent des traitements
parallélisés"
root="stanford-postagger-full-2020-11-17"
stf = StanfordPOSTagger(root+'/models/french-
ud.tagger',root+"/stanford-postagger.jar",encoding='utf8')
tokens = nltk.word_tokenize(data)
print(stf.tag(tokens))
```

#8. Analyse Sémantique

Un mot peut avoir plusieurs significations selon son contexte (les mots voisins et le rôle grammaticale). Par exemple, le mot anglais « break » possède 75 sens. Chose qui montre l'importance de la désambiguïsation lors de l'analyse d'un texte. Ci-dessous un extrait de la récupération des différents sens du mot « break » avec leurs annotations grammaticales.

```
from nltk.corpus import wordnet
for synset in wordnet.synsets('break')[:10]:
    print(">>>",synset.definition())
```

```
>>> some abrupt occurrence that interrupts an ongoing activity
>>> an unexpected piece of good luck
>>> (geology) a crack in the earth's crust resulting from the
displacement of one side with respect to the other
>>> a personal or social separation (as between opposing factions)
>>> a pause from doing something (as work)
>>> the act of breaking something
>>> a time interval during which there is a temporary cessation of
something
>>> breaking of hard tissue such as bone
>>> the occurrence of breaking
>>> an abrupt change in the tone or register of the voice (as at
puberty or due to emotion)
```

Pour un synset bien determiné on peut recuperer la liste des termes qui partage le même sens (La même description du sens):

```
from nltk.corpus import wordnet
seynsets= wordnet.synsets('break')

for synset in seynsets[:10]:
    print(synset.lemmas())
```

```
[Lemma('interruption.n.02.interruption'),
Lemma('interruption.n.02.break')]
[Lemma('break.n.02.break'), Lemma('break.n.02.good_luck'),
Lemma('break.n.02.happy_chance')]
[Lemma('fault.n.04.fault'), Lemma('fault.n.04.faulting'),
Lemma('fault.n.04.geological_fault'), Lemma('fault.n.04.shift'),
```

```
Lemma('fault.n.04.fracture'), Lemma('fault.n.04.break')]
[Lemma('rupture.n.02.rupture'), Lemma('rupture.n.02.breach'),
Lemma('rupture.n.02.break'), Lemma('rupture.n.02.severance'),
Lemma('rupture.n.02.rift'), Lemma('rupture.n.02.falling_out')]
[Lemma('respite.n.02.respite'), Lemma('respite.n.02.recess'),
Lemma('respite.n.02.break'), Lemma('respite.n.02.time_out')]
[Lemma('breakage.n.03.breakage'), Lemma('breakage.n.03.break'),
Lemma('breakage.n.03.breaking')]
[Lemma('pause.n.01.pause'), Lemma('pause.n.01.intermission'),
Lemma('pause.n.01.break'), Lemma('pause.n.01.interruption'),
Lemma('pause.n.01.suspension')]
[Lemma('fracture.n.01.fracture'), Lemma('fracture.n.01.break')]
[Lemma('break.n.09.break')]
[Lemma('break.n.10.break')]
```

On peut même récuperer le terme correspedant dans une autre langue

```
from nltk.corpus import wordnet
seynsets= wordnet.synsets('break')

#Francais
for synset in seynsets[:10]:
    print(synset.lemmas(lang='fra'))

#Arabe
for synset in seynsets[:10]:
    print(synset.lemmas(lang='arb'))
```

```
[Lemma('interruption.n.02.interruption')]
[Lemma('break.n.02.casser')]
[Lemma('fault.n.04.casser'), Lemma('fault.n.04.cassure'),
Lemma('fault.n.04.fracture')]
[Lemma('rupture.n.02.briser'), Lemma('rupture.n.02.chute'),
Lemma('rupture.n.02.rift'), Lemma('rupture.n.02.rompre')]
[Lemma('respite.n.02.casser'), Lemma('respite.n.02.pause'),
Lemma('respite.n.02.repos'), Lemma('respite.n.02.trêve')]
[Lemma('breakage.n.03.cassure')]
[Lemma('pause.n.01.intermission'), Lemma('pause.n.01.interruption'),
Lemma('pause.n.01.pause'), Lemma('pause.n.01.repos'),
Lemma('pause.n.01.trêve')]
[Lemma('fracture.n.01.casser'), Lemma('fracture.n.01.fracture')]
[Lemma('break.n.09.casser')]
[Lemma('break.n.10.casser')]
[Lemma('interruption.n.02.توقُّف')]
[]
[]
[]
[]
[Lemma('breakage.n.03.تحْطِيم'), Lemma('breakage.n.03.تكْسِير'),
```

```
Lemma('breakage.n.03.كسْر')]
[Lemma('pause.n.01.إرْجاء'), Lemma('pause.n.01.اِسْتِراحة'),
Lemma('pause.n.01.اِيقاف'), Lemma('pause.n.01.تعْلِيق'), Lemma('pause.n.01.
توقُّف')]
[Lemma('fracture.n.01.كسْر')]
[]
[]
```

pour la liste des langues dsiponibles executer: sorted(wn.langs())

La bibliothèque nltk offre à travers le module **wsd** la possibilité de détecter le sens d'un mot en fonction de son contexte. A cette fin, l'algorithme **Lesk** est utilisé pour réaliser une désambiguïsation du sens d'un mot en retournant le sens qui a permis d'avoir le plus grand nombre de termes en intersection avec le contexte du mot pour lequel on est en train de chercher le sens exact. L'algorithme ne retourne aucun sens s'il n'arrive pas à réaliser la désambiguïsation.

```
from nltk.wsd import lesk
from nltk.tokenize import word_tokenize

context= word_tokenize("I've just finished the first step of the
competition. I need a little break to catch my breath")
synset=lesk(context, 'break','n')
print(synset.definition())
```

```
(geology) a crack in the earth's crust resulting from the displacement
of one side with respect to the other
```

L'exemple ci-dessus montre que l'algorithme n'est pas assez performant. D'autres algorithmes peuvent être utilisés en se basant sur les bibliothèques baseline, pywsd ou spaCy. Ci-dessous un autre exemple avec la bibliotheque pywsd.

```
# !pip install pywsd
# pip install -U pywsd
#pip install wn==0.0.22
from pywsd.lesk import simple_lesk
sent = "I've just finished the first step of the competition. I need a
little break to catch my breath"
ambiguous = 'break'
answer = simple_lesk(sent, ambiguous, pos='n')
print (answer)
print (answer.definition())
```

```
Warming up PyWSD (takes ~10 secs)...
```

```
Synset('rupture.n.02')
a personal or social separation (as between opposing factions)
```

```
took 3.2237613201141357 secs.
```

#9. Analyse syntaxique

L'objectif de cette section est d'analyser la structure grammaticale des phrases au lieu de se focaliser d'une manière individuelle sur les mots les composant. Nous nous contenetant dans un premier temps de l'approche grammaticale pour realiser l'inference de la structure arborescente d'une phrase.

Il existe plusieurs bibliotehques permettant de reéaliser cette tâche. ci-dessous deux exemples avec les bibliotheque stanford et spacy

Stanford

```
#import os
from nltk.parse import stanford
!os.environ['STANFORD_PARSER'] = 'stanford-parser-full-2020-11-17'
!os.environ['STANFORD_MODELS'] = 'stanford-parser-full-2020-11-17'

parser = stanford.StanfordParser(model_path="stanford-parser-full-
2020-11-17/stanford-parser-4.2.0-models/edu/stanford/nlp/models/
lexparser/englishPCFG.ser.gz")
sentences = parser.raw_parse_sents(("Hello, My name is Melroy.", "What
is your name?"))


#Formatted

for line in sentences:
    for sentence in line:
        print(sentence)


'''
# GUI
for line in sentences:
    for sentence in line:
        sentence.draw()

'''
```

SpaCy

spaCy une bibliothèque de la NLP qui est très puissante (elle est orientée production, non uniquement pour la recherche ou l'apprentissage de la NLP), totalement écrite python, gratuite et libre. Par défaut, lorsqu'on fait appel au module **nlp** de spaCy, les opérations suivantes sont exécutées : Segmentation, pos-tagging, analyse syntaxique, NER (Named Entity Recognition), etc. Un objet Doc est retourné à l'issue de toutes les opérations et qui encapsule tous les resultats de l'analyse.

L'exemple ci-dessous montre comment extraire les différentes informations à partir d'un objet Doc.

```
# !python -m spacy download fr_core_news_sm

import spacy
nlp = spacy.load('fr_core_news_sm')
doc = nlp("Le big data  « grosses données » en anglais,les mégadonnées
ou les données massives, " \
     "désigne les ressources d’informations dont les caractéristiques
en termes de volume," \
     " de vélocité et de variété imposent l’utilisation de
technologies et de méthodes analytiques " \
     "particulières pour générer de la valeur, et qui dépassent en
général les capacités " \
     "d'une seule et unique machine et nécessitent des traitements
parallélisés")

for token in doc:
    print("/token:",token.text, "/lemma:",token.lemma_, token.shape_,
token.is_alpha, token.is_stop,"/POS:", token.tag_, "/PARS:",
token.head, token.dep_, "/NER:", token.ent_iob_, token.ent_type_)
```

```
/token: Le /lemma: le Xx True True /POS: DET /PARS: big det /NER: O
/token: big /lemma: big xxx True False /POS: PRON /PARS: data nsubj
/NER: O
/token: data /lemma: dater xxxx True False /POS: VERB /PARS: data ROOT
/NER: O
/token:   /lemma:     False False /POS: SPACE /PARS: data dep /NER: O
/token: « /lemma: « « False False /POS: NUM /PARS: data xcomp /NER: O
/token: grosses /lemma: grosse xxxx True False /POS: NOUN /PARS:
grosses ROOT /NER: O
/token: données /lemma: donnée xxxx True False /POS: NOUN /PARS:
grosses amod /NER: O
/token: » /lemma: » » False False /POS: PUNCT /PARS: grosses punct
/NER: O
/token: en /lemma: en xx True True /POS: ADP /PARS: anglais case /NER:
O
/token: anglais /lemma: anglais xxxx True False /POS: NOUN /PARS:
grosses nmod /NER: O
/token: , /lemma: , , False False /POS: PUNCT /PARS: désigne punct
/NER: O
/token: les /lemma: le xxx True True /POS: DET /PARS: mégadonnées
det /NER: O
/token: mégadonnées /lemma: mégadonnée xxxx True False /POS: NOUN
/PARS: désigne nsubj /NER: O
/token: ou /lemma: ou xx True True /POS: CCONJ /PARS: données cc /NER:
O
/token: les /lemma: le xxx True True /POS: DET /PARS: données det
/NER: O
/token: données /lemma: donnée xxxx True False /POS: NOUN /PARS:
mégadonnées conj /NER: O
/token: massives /lemma: massif xxxx True False /POS: ADJ /PARS:
```

```
données amod /NER: O
/token: , /lemma: , , False False /POS: PUNCT /PARS: désigne punct
/NER: O
/token: désigne /lemma: désigner xxxx True False /POS: VERB /PARS:
désigne ROOT /NER: O
/token: les /lemma: le xxx True True /POS: DET /PARS: ressources
det /NER: O
/token: ressources /lemma: ressource xxxx True False /POS: NOUN /PARS:
désigne obj /NER: O
/token: d’ /lemma: d’ x’ False True /POS: ADJ /PARS: désigne dep /NER:
O
/token: informations /lemma: information xxxx True False /POS: NOUN
/PARS: désigne obj /NER: O
/token: dont /lemma: dont xxxx True True /POS: PRON /PARS:
caractéristiques nmod /NER: O
/token: les /lemma: le xxx True True /POS: DET /PARS: caractéristiques
det /NER: O
/token: caractéristiques /lemma: caractéristique xxxx True False /POS:
NOUN /PARS: imposent nsubj /NER: O
/token: en /lemma: en xx True True /POS: ADP /PARS: termes case /NER:
O
/token: termes /lemma: terme xxxx True False /POS: NOUN /PARS:
caractéristiques nmod /NER: O
/token: de /lemma: de xx True True /POS: ADP /PARS: volume case /NER:
O
/token: volume /lemma: volume xxxx True False /POS: NOUN /PARS: termes
nmod /NER: O
/token: , /lemma: , , False False /POS: PUNCT /PARS: vélocité punct
/NER: O
/token: de /lemma: de xx True True /POS: ADP /PARS: vélocité case
/NER: O
/token: vélocité /lemma: vélocité xxxx True False /POS: NOUN /PARS:
termes conj /NER: O
/token: et /lemma: et xx True True /POS: CCONJ /PARS: variété cc /NER:
O
/token: de /lemma: de xx True True /POS: ADP /PARS: variété case /NER:
O
/token: variété /lemma: variété xxxx True False /POS: AUX /PARS:
termes conj /NER: O
/token: imposent /lemma: imposer xxxx True False /POS: VERB /PARS:
informations acl:relcl /NER: O
/token: l’ /lemma: l’ x’ False True /POS: NOUN /PARS: utilisation
det /NER: O
/token: utilisation /lemma: utilisation xxxx True False /POS: NOUN
/PARS: imposent obj /NER: O
/token: de /lemma: de xx True True /POS: ADP /PARS: technologies
case /NER: O
/token: technologies /lemma: technologie xxxx True False /POS: NOUN
/PARS: utilisation nmod /NER: O
```

```
/token: et /lemma: et xx True True /POS: CCONJ /PARS: méthodes cc
/NER: O
/token: de /lemma: de xx True True /POS: ADP /PARS: méthodes case
/NER: O
/token: méthodes /lemma: méthode xxxx True False /POS: NOUN /PARS:
technologies conj /NER: O
/token: analytiques /lemma: analytique xxxx True False /POS: ADJ
/PARS: méthodes amod /NER: O
/token: particulières /lemma: particulier xxxx True False /POS: ADJ
/PARS: méthodes amod /NER: O
/token: pour /lemma: pour xxxx True True /POS: ADP /PARS: générer mark
/NER: O
/token: générer /lemma: générer xxxx True False /POS: VERB /PARS:
imposent advcl /NER: O
/token: de /lemma: de xx True True /POS: ADP /PARS: valeur case /NER:
O
/token: la /lemma: le xx True True /POS: DET /PARS: valeur det /NER: O

/token: valeur /lemma: valeur xxxx True False /POS: NOUN /PARS:
générer obl:arg /NER: O
/token: , /lemma: , , False False /POS: PUNCT /PARS: imposent punct
/NER: O
/token: et /lemma: et xx True True /POS: CCONJ /PARS: dépassent cc
/NER: O
/token: qui /lemma: qui xxx True True /POS: PRON /PARS: dépassent
nsubj /NER: O
/token: dépassent /lemma: dépasser xxxx True False /POS: VERB /PARS:
imposent conj /NER: O
/token: en /lemma: en xx True True /POS: ADP /PARS: général case /NER:
O
/token: général /lemma: général xxxx True False /POS: NOUN /PARS:
dépassent obl:mod /NER: O
/token: les /lemma: le xxx True True /POS: DET /PARS: capacités det
/NER: O
/token: capacités /lemma: capacité xxxx True False /POS: NOUN /PARS:
dépassent obj /NER: O
/token: d' /lemma: de x' False True /POS: ADP /PARS: seule case /NER:
O
/token: une /lemma: un xxx True True /POS: DET /PARS: seule det /NER:
O
/token: seule /lemma: seul xxxx True True /POS: ADJ /PARS: capacités
amod /NER: O
/token: et /lemma: et xx True True /POS: CCONJ /PARS: machine cc /NER:
O
/token: unique /lemma: unique xxxx True False /POS: ADJ /PARS: machine
amod /NER: O
/token: machine /lemma: machine xxxx True False /POS: NOUN /PARS:
seule conj /NER: O
/token: et /lemma: et xx True True /POS: CCONJ /PARS: nécessitent
```

```
cc /NER: O
/token: nécessitent /lemma: nécessiter xxxx True False /POS: VERB
/PARS: dépassent conj /NER: O
/token: des /lemma: un xxx True True /POS: DET /PARS: traitements
det /NER: O
/token: traitements /lemma: traitement xxxx True False /POS: NOUN
/PARS: nécessitent obj /NER: O
/token: parallélisés /lemma: paralléliser xxxx True False /POS:
VERB /PARS: traitements acl /NER: O
```

Pour récuperer L'arbre de dependance syntaxique evec spaCy

```
# !python -m spacy download en_core_web_sm
import spacy
from spacy import displacy


nlp = spacy.load("en_core_web_sm")
doc = nlp("Hello, My name is Melroy. What is your name?")

#Visualisation 1
print ("{:<15} | {:<8} | {:<15} |{:<10} |
{:<20}".format('Token','Relation','Head','POS', 'Children'))
print ("-" * 70)
for token in doc:
  # Print the token, dependency nature, head and all dependents of the
token
  print ("{:<15} | {:<8} | {:<15} |{:<10} | {:<20}"
         .format(str(token.text), str(token.dep_),
str(token.head.text), str(token.head.pos_), str([child for child in
token.children])))


#Visualisation 2 (graphique)
displacy.render(doc, style='dep', jupyter=True, options={'distance':
120})
```

```
Token           | Relation | Head            |POS        | Children

----------------------------------------------------------------------
Hello           | intj     | is              |AUX        | []

,               | punct    | is              |AUX        | []

My              | poss     | name            |NOUN       | []

name            | nsubj    | is              |AUX        | [My]

is              | ROOT     | is              |AUX        | [Hello, ,,
name, Melroy, .]
```

```
Melroy          | attr     | is              |AUX        | []

.               | punct    | is              |AUX        | []

What            | attr     | is              |AUX        | []

is              | ROOT     | is              |AUX        | [What,
name, ?]
your            | poss     | name            |NOUN       | []

name            | nsubj    | is              |AUX        | [your]

?               | punct    | is              |AUX        | []

<IPython.core.display.HTML object>
```

10. Exercices :

10.1 Exercice 1: Traduction automatique

On desire assister l'utilisateur pendant la traduction de l'anglais vers le francais.

- Constituer le contexte du document en recuperant tous les termes sigfificatifs
- Découper le texte en des phrases simples et recuperer les tags de leurs mots.
- Pour chaque phrase récuperer le sens exacte de chaque terme en se basant sur leurs Tags et leur contexts
- Récuperer les termes correspendant en francais
- Pour chaque phrase afficher à l'utilisateur les propostions de traduction pour les nom, les adjectifs et les verbes

Proposition - Exercice 01

```
import spacy
from nltk.corpus import wordnet as wn
from nltk.wsd import lesk

# Load spaCy's English model
nlp = spacy.load("en_core_web_sm")

text = """Inheritance is a basic concept of Object-Oriented
Programming. It allows a class to use properties and methods of
another class."""

def extract_significant_terms(doc):
    """
    Extract all significant terms from the document.
    """
```

```
    significant_terms = [token.text for token in doc if not
token.is_stop and not token.is_punct]
    return significant_terms

def split_and_tag_sentences(doc):
    """
    Split the text into simple sentences and retrieve the POS tags for
each word.
    """
    sentences = []
    for sent in doc.sents:
        tagged_words = [(token.text, token.pos_) for token in sent]
        sentences.append(tagged_words)
    return sentences

def retrieve_exact_meaning_synset(word, sentence):
    """
    Use WordNet to retrieve the exact sense (meaning) of the word in
the context of the sentence.
    """
    synset = lesk(sentence, word)
    if synset:
        return synset
    return None

def translate_with_synset(word, sentence):
    """
    Retrieve French translation of the word using its synset.
    """
    synset = retrieve_exact_meaning_synset(word, sentence)
    if synset:
        translations = synset.lemmas('fra')
        if translations:
            return translations[0].name()
    return word

def display_translation_suggestions(tagged_sentence, sentence):
    """
    Display translation suggestions for nouns, adjectives, and verbs.
    """
    suggestions = []
    for word, pos in tagged_sentence:
        if pos in ["NOUN", "VERB", "ADJ"]:
            french_term = translate_with_synset(word, sentence)
            suggestions.append((word, french_term, pos))
    return suggestions

doc = nlp(text)

significant_terms = extract_significant_terms(doc)
```

```
print("Significant Terms from the Document (Context):",
significant_terms)

tagged_sentences = split_and_tag_sentences(doc)
print("\nTagged Sentences:")
for sentence in tagged_sentences:
    print(sentence)

print("\nTranslation Suggestions (For Nouns, Adjectives, and Verbs):")
for sentence in doc.sents:
    tagged_sentence = [(token.text, token.pos_) for token in sentence]
    suggestions = display_translation_suggestions(tagged_sentence,
sentence)
    for original, french, pos in suggestions:
        print(f"Original: {original} ({pos}) -> Suggested Translation:
{french}")
```

```
Significant Terms from the Document (Context): ['Inheritance',
'basic', 'concept', 'Object', 'Oriented', 'Programming', 'allows',
'class', 'use', 'properties', 'methods', 'class']

Tagged Sentences:
[('Inheritance', 'NOUN'), ('is', 'AUX'), ('a', 'DET'), ('basic',
'ADJ'), ('concept', 'NOUN'), ('of', 'ADP'), ('Object', 'NOUN'), ('-',
'PUNCT'), ('Oriented', 'VERB'), ('Programming', 'PROPN'), ('.',
'PUNCT')]
[('It', 'PRON'), ('allows', 'VERB'), ('a', 'DET'), ('class', 'NOUN'),
('to', 'PART'), ('use', 'VERB'), ('properties', 'NOUN'), ('and',
'CCONJ'), ('methods', 'NOUN'), ('of', 'ADP'), ('another', 'DET'),
('class', 'NOUN'), ('.', 'PUNCT')]

Translation Suggestions (For Nouns, Adjectives, and Verbs):
Original: Inheritance (NOUN) -> Suggested Translation: héritage
Original: basic (ADJ) -> Suggested Translation: basique
Original: concept (NOUN) -> Suggested Translation: concept
Original: Object (NOUN) -> Suggested Translation: chose
Original: Oriented (VERB) -> Suggested Translation: adapter
Original: allows (VERB) -> Suggested Translation: autoriser
Original: class (NOUN) -> Suggested Translation: classe
Original: use (VERB) -> Suggested Translation: de
Original: properties (NOUN) -> Suggested Translation: accessoire
Original: methods (NOUN) -> Suggested Translation: methods
Original: class (NOUN) -> Suggested Translation: classe
```

10.2 Exercice 2: Detection du plagiarisme

L'objectif de cet exercice est de détecter le pélagianisme à partir de wikipedia pendant la préparation des réponses à un certain nombre de questions sur des connaissances en informatique. Le dataset utilisé peut-être récupéré à partir du lien suivant :Cliquer ICI

Pour ce faire, nous nous basant sur le calcul des similarités entre les réponses des candidats et les définitions exactes trouvées sur Wikipédia. Deux méthodes de calcul de similarité sont à utiliser, à savoir, la similarité syntaxique (orientée caractères) et la similarité sémantique.

Exploration du corpus

Le corpus utilisé dans cette exercice était realisé dans le cadre d'un travail de recherche "Clough, P., Stevenson, M. Developing a corpus of plagiarised short answers. Lang Resources & Evaluation 45, 5–24 (2011). https://doi.org/10.1007/s10579-009-9112-1"

Le corpus se compose de plusieurs fichiers texte dont les carateristiques sont decrites dans corpus-final09.xls.

Chaque fichier est associé à une tache (tasks a-e) et à un type de plagiarisme:

- cut: du copier coller à partir du texte original.
- light: quelquie extrait du texte original avec queklques reformulations.
- heavy: du copier collller vaec reformulation.
- non: pas de copier coller.
- orig: texte origional.

Similarité Syntaxique

Pour la similarité syntaxique entre des documents courts (des phrases) on peut recrorir à l'utilisation de l'un des algorithmes suivants:

- Longest Common Sequence (LCS)
- Set features
- Word Order Similarity
- n-gram sentences
- Jaro-Winkler
- ...
- Recuperer le dataset du plagiarisme?
- Réaliser les différentes tâches de prétraitement?
- Calculer les similarités syntaxiques entre les réponses des étudiants et les reponses originales.

```
#glob lib
!pip install glob2
```

```
Requirement already satisfied: glob2 in c:\users\dscon\anaconda3\lib\
site-packages (0.7)
```

Similarité Sémantique

WordNet est une base de données lexicale qui comporte des concepts (termes) classifiés et reliés les uns aux autres à travers des realtions semantiques

La composante principale de wordNet est le synset (synonym set) tel que chacun contient plusieurs mots qui partagent le même sens (des lemmas). Egalement, un mot peut appartenir à plusieurs synsets à la foix.

l'exemple suivant montre comment recuperer les synsets d'un mot et comment recuperer ses synonymes pour un sens particuliers

```
from nltk.corpus import wordnet as wn
computer_synsets = wn.synsets("computer")
print("Computer sens in wordNet:")
i=0
for sense in computer_synsets:
    print(" \t Sens :", i)
    print(" \t\t Sens definition: "+sense.definition())
    lemmas = [l.name() for l in sense.lemmas()]
    print("\t\t Lemmas for sense :" +str(lemmas))
    i=i+1
```

```
Computer sens in wordNet:
      Sens : 0
           Sens definition: a machine for performing calculations
automatically
           Lemmas for sense :['computer', 'computing_machine',
'computing_device', 'data_processor', 'electronic_computer',
'information_processing_system']
      Sens : 1
           Sens definition: an expert at calculation (or at operating
calculating machines)
           Lemmas for sense :['calculator', 'reckoner', 'figurer',
'estimator', 'computer']
```

Pour la similarité sémantique, la bibliothèque nltk et à travers le module wordnet permet de mesurer la distance ou la similarité sémantique entre les sens des mots. Ainsi, en récupérant les sens synset1 et synset2 de deux mots quelconques plusieurs façons sont possibles pour calculer leur similarité:

- synset1.path_similarity(synset2) : retourne leur ordre de similarité sous forme d'une valeur numérique entre 0 et 1 en se basant sur le plus court chemin qui relie les deux sens dans l'arborescence de wordnet.
- synset1.lch_similarity(synset2): qui se base sur l'algorithme Leacock-Chodorow
- Synset1.wup_similarity(synset2): qui se base sur l'algorithme Wu-Palmer
- synset1.res_similarity(synset2, ic): qui se base sur l'algorithme Resnik:
- synset1.jcn_similarity(synset2, ic): qui se base sur l'algorithme Jiang-Conrath
- synset1.lin_similarity(synset2, ic): qui se base sur l'algorithme Lin

l'exemple suivant montre comment calculer les similarité entres les sens des termes computer et device en se basant sur les metriques Leacock-Chodorow et Wu-Palmer

```
from nltk.corpus import wordnet as wn
import pandas as pd
import numpy as np
computer_synsets = wn.synsets("computer")
device_synsets = wn.synsets("device")
lch=[]
wup=[]


for s1 in computer_synsets:
    for s2 in device_synsets:
        lch.append(s1.lch_similarity(s2))
        wup.append(s1.wup_similarity(s2))

pd.DataFrame([lch,wup],["lch","wup"])
```

```
            0         1         2         3         4         5
6  \
lch  2.538974  1.072637  0.693147  1.558145  1.440362  1.440362
1.335001
wup  0.875000  0.142857  0.100000  0.588235  0.555556  0.500000
0.181818

            7         8         9
lch  0.864997  1.335001  1.239691
wup  0.117647  0.470588  0.444444
```

Souvent on aura besoin de recuperer les sens exactes des termes dans leurs contextes afin mesurer leurs similarité d'une manière plus precise

```
from nltk.wsd import lesk
from nltk.tokenize import word_tokenize
def WSD(word, doc):
    context= word_tokenize(doc)
    sens=lesk(context, word)
    return sens

doc1='Computer science is the study of computers and computing
concepts. It includes both hardware and software, as well as
networking and the Internet'
doc2='Computer science is the science that deals with the theory and
methods of processing information in digital computers, the design of
computer hardware and software, and the applications of computers.'


print(WSD("Computer", doc1).definition())
print(WSD("Computer", doc2).definition())
```

```
a machine for performing calculations automatically
a machine for performing calculations automatically
```

Pour calculer la distance semantique entre deux documents, on aura besoin de calculer les similarités semantiques entre leurs mots deux à deux ou utiliser par example la distance de Hausdorff ou l'indic de Jaccard.

- Defnir la fonction SemanticDistanceDocs(doc1,doc2) qui permet de calculer la distance semantique totale entre deux documents texte?

```
from nltk.corpus import wordnet as wn
from nltk.tokenize import word_tokenize

doc1='Computer science is the study of computers and computing
concepts. It includes both hardware and software, as well as
networking and the Internet'
doc2='Computer science is the science that deals with the theory and
methods of processing information in digital computers, the design of
computer hardware and software, and the applications of computers.'

def WSD(word, sentence):
    return lesk(sentence, word)

def synset_similarity(synset1, synset2):
    if synset1 is None or synset2 is None:
        return 0  # Aucun synset trouvé pour l'un des mots, pas de
similarité possible
    return synset1.wup_similarity(synset2) or 0  # Wu-Palmer
Similarity


def SemanticDistanceDocs(doc1, doc2):
    # Tokenisation des documents
    words_doc1 = word_tokenize(doc1)
    words_doc2 = word_tokenize(doc2)

    # Désambiguïsation des sens pour chaque mot dans les deux
documents
    senses_doc1 = [WSD(word, doc1) for word in words_doc1]
    senses_doc2 = [WSD(word, doc2) for word in words_doc2]

    # Calculer la similarité entre tous les sens (synsets) des deux
documents
    total_similarity = 0
    count = 0
    for sense1 in senses_doc1:
        for sense2 in senses_doc2:
            similarity = synset_similarity(sense1, sense2)
            total_similarity += similarity
            count += 1


    # Si aucun synset n'a pu être trouvé, retourner 0
    if count == 0:
```

```
        return 0

    # Retourner la similarité moyenne entre les synsets des deux
documents
    average_similarity = total_similarity / count
    return 1 - average_similarity  # La distance est l'inverse de la
similarité


SemanticDistanceDocs(doc1, doc2)
```

```
0.9111448876638121
```

- Calculer les similarités sémantiqueq entre les réponses des étudiants et les définitions trouvées sur wikipedia? considerer par example que:
  - cut: correspond à une similarité entre 75% et 100%
  - heavy: correspond à une similarité entre 50% et 75%
  - light: correspond à une similarité entre 25% et 50%
  - Non: correspond à une similarité entre 0% et 25%

```
import glob
def NLP(text):
    lemmes = [lemmatizer.lemmatize(word.lower()) for word in
word_tokenize(text) if (not word in set(stopwords.words('english'))
and  word.isalpha())]
    return lemmes

for file in glob.glob("*user.txt"):
    with open(file, 'r') as f:
        text=f.read()
        task=file.len

# !pip install xlrd==2.0.1
# !pip install openpyxl
```

10.2 Exercice 2: Detection du plagiarisme

1.a. Data Preprocessing

```
import re
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words('english'))
```

```
def preprocess_text(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = re.sub(r'\d+', '', text)
    words = word_tokenize(text)
    words = [word for word in words if word not in stop_words]
    return ' '.join(words)
```

2. Calcul de similarités Syntaxiques

```
import glob, os
import pandas as pd

path = 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Original/
*.txt'
def txts_file_to_df(path):
    file_paths = glob.glob(path)
    print(file_paths)
    data = []
    for filepath in file_paths:
        try:
            with open(filepath, 'r', encoding='utf-8') as file:
                content = file.read()
        except UnicodeDecodeError:
            with open(filepath, 'r', encoding='ISO-8859-1') as file:
                content = file.read()
        filename = os.path.basename(filepath)
        data.append({
            "File": filename,
            'Content': content,
            'Task': filename.split('task')[-1].split('.')[0]
        })
    df = pd.DataFrame(data)
    return df

response_files = txts_file_to_df(path)
```

```
['c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Original\\
orig_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Original\\
orig_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Original\\
orig_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Original\\
orig_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Original\\
orig_taske.txt']
```

```
path = 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users/*.txt
'
users_files = txts_file_to_df(path)

def getName(text):
    return str(text).split('_')[0]

users_files['User'] = users_files['File'].apply(getName)
```

```
['c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pA_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pA_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pA_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pA_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pA_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pB_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pB_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pB_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pB_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pB_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pC_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pC_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pC_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pC_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pC_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pD_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pD_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pD_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pD_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
```

```
g0pD_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pE_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pE_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pE_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pE_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g0pE_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pA_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pA_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pA_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pA_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pA_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pB_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pB_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pB_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pB_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pB_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pD_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pD_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pD_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pD_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g1pD_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pA_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pA_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pA_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pA_taskd.txt', 'c:/Users/dscon/Documents/COURS
```

```
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pA_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pB_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pB_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pB_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pB_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pB_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pC_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pC_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pC_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pC_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pC_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pE_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pE_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pE_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pE_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g2pE_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pA_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pA_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pA_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pA_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pA_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pB_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pB_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pB_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
```

```
g3pB_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pB_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pC_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pC_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pC_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pC_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g3pC_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pB_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pB_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pB_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pB_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pB_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pC_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pC_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pC_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pC_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pC_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pD_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pD_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pD_taskc.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pD_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pD_taske.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pE_taska.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pE_taskb.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pE_taskc.txt', 'c:/Users/dscon/Documents/COURS
```

```
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pE_taskd.txt', 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/plagitat/Users\\
g4pE_taske.txt']
```

```
response_files
```

```
             File                                                Content
Task
0  orig_taska.txt  In object-oriented programming, inheritance is...
a
1  orig_taskb.txt  PageRank is a link analysis algorithm used by ...
b
2  orig_taskc.txt  Vector space model (or term vector model) is a...
c
3  orig_taskd.txt  In probability theory, Bayes' theorem (often c...
d
4  orig_taske.txt  In mathematics and computer science, dynamic p...
e
```

```
# change column name content to Response
response_files.rename(columns={'Content': 'OResponse'}, inplace=True)
users_files.rename(columns={'Content': 'UResponse'}, inplace=True)
```

```
users_files
```

```
              File                                           UResponse
Task  \
0   g0pA_taska.txt  Inheritance is a basic concept of Object-Orien...
a
1   g0pA_taskb.txt  PageRank is a link analysis algorithm used by ...
b
2   g0pA_taskc.txt  The vector space model (also called, term vect...
c
3   g0pA_taskd.txt  Bayes’ theorem was names after Rev Thomas Baye...
d
4   g0pA_taske.txt  Dynamic Programming is an algorithm design tec...
e
..             ...                                              ...
...
90  g4pE_taska.txt  Object oriented programming is a  style of pro...
a
91  g4pE_taskb.txt  PageRankalgorithm is also known as link analys...
b
92  g4pE_taskc.txt  The definition of term depends on the applicat...
c
93  g4pE_taskd.txt  "Bayes' Theorem" or "Bayes' Rule", or somethin...
d
94  g4pE_taske.txt   Dynamic programming is a method for efficient...
e
```

```
    User
0   g0pA
1   g0pA
2   g0pA
3   g0pA
4   g0pA
..   ...
90  g4pE
91  g4pE
92  g4pE
93  g4pE
94  g4pE

[95 rows x 4 columns]
```

```
# Load data
excel_path = 'c:/Users/dscon/Documents/COURS
UM6P/S3/TEXT-MINING/Corpus_Plagiat/Corpus_Plagiat/corpus-final09.xls'
sheet_name = 'File list'
decision_df = pd.read_excel(excel_path, sheet_name=sheet_name)

# Extract User and Task from filename
decision_df['User'] = decision_df['File'].apply(getName)
decision_df
```

```
              File  Group Person Task Category Native English
Knowledge  \
0   g0pA_taska.txt      0      A    a      non         native
1
1   g0pA_taskb.txt      0      A    b      cut         native
4
2   g0pA_taskc.txt      0      A    c    light         native
5
3   g0pA_taskd.txt      0      A    d    heavy         native
3
4   g0pA_taske.txt      0      A    e      non         native
4
..             ...    ...    ...  ...      ...            ...         .
..
90  g4pE_taska.txt      4      E    a    heavy     non-native
1
91  g4pE_taskb.txt      4      E    b    light     non-native
3
92  g4pE_taskc.txt      4      E    c      cut     non-native
4
93  g4pE_taskd.txt      4      E    d      non     non-native
4
94  g4pE_taske.txt      4      E    e      non     non-native
4
```

```
    Difficulty  User
0            1  g0pA
1            3  g0pA
2            3  g0pA
3            4  g0pA
4            3  g0pA
..         ...   ...
90           2  g4pE
91           2  g4pE
92           2  g4pE
93           4  g4pE
94           5  g4pE

[95 rows x 9 columns]
```

```
# merge the two dataframes on the 'Task' column
merged_df = pd.merge(response_files[['OResponse', 'Task']],
users_files, on=['Task'], how='inner')
merged_df = pd.merge(merged_df, decision_df, on=['Task', 'User'],
how='inner')
merged_df.to_csv('merged_df.csv', index=False)

new_df = merged_df[['User', 'Task', 'OResponse', 'UResponse',
'Category']]
new_df
```

```
    User Task                                          OResponse  \
0   g0pA    a  In object-oriented programming, inheritance is...
1   g0pB    a  In object-oriented programming, inheritance is...
2   g0pC    a  In object-oriented programming, inheritance is...
3   g0pD    a  In object-oriented programming, inheritance is...
4   g0pE    a  In object-oriented programming, inheritance is...
..   ...  ...                                                ...
90  g3pC    e  In mathematics and computer science, dynamic p...
91  g4pB    e  In mathematics and computer science, dynamic p...
92  g4pC    e  In mathematics and computer science, dynamic p...
93  g4pD    e  In mathematics and computer science, dynamic p...
94  g4pE    e  In mathematics and computer science, dynamic p...

                                          UResponse Category
0   Inheritance is a basic concept of Object-Orien...      non
1   Inheritance is a basic concept in object orien...      non
2   inheritance in object oriented programming is ...    heavy
3   Inheritance in object oriented programming is ...      cut
4   In object-oriented programming, inheritance is...    light
..                                                ...      ...
90  In computer science and mathematics, dynamic p...    light
91  In mathematics and computer science, dynamic p...      cut
92  In mathematics and computer science, dynamic p...    light
```

```
93  Dynamic programming is a method of providing s...    heavy
94   Dynamic programming is a method for efficient...      non

[95 rows x 5 columns]
```

```python
import re
import string
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from difflib import SequenceMatcher

# Prétraitement du texte
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = re.sub(r'\d+', '', text)
    words = word_tokenize(text)
    words = [word for word in words if word not in stop_words]
    return ' '.join(words)


def jaro_winkler_similarity(str1, str2):
    return SequenceMatcher(None, str1, str2).ratio()

def ngram_similarity(str1, str2, n=2):
    str1 = str1.split()
    str2 = str2.split()
    ngrams1 = {tuple(str1[i:i+n]) for i in range(len(str1) - n + 1)}
    ngrams2 = {tuple(str2[i:i+n]) for i in range(len(str2) - n + 1)}
    intersection = ngrams1.intersection(ngrams2)
    return len(intersection) / float(len(ngrams1.union(ngrams2)))

def set_features_similarity(str1, str2):
    set1 = set(str1.split())
    set2 = set(str2.split())
    intersection = set1.intersection(set2)
    return len(intersection) / float(len(set1.union(set2)))

def word_order_similarity(str1, str2):
    words1 = str1.split()
    words2 = str2.split()
    matching_words = sum(1 for word in words1 if word in words2)
    return matching_words / float(max(len(words1), len(words2)))

def apply_syntaxic_similarity(df, algorithms):
    # Pour chaque algorithme, ajouter une colonne vide dans le
DataFrame
```

```
    for algo_name in algorithms.keys():
        df[algo_name] = 0.0  # Initialiser chaque colonne avec des
scores à 0.0

    # Appliquer chaque algorithme de similarité aux lignes du
DataFrame
    for index, row in df.iterrows():
        response = preprocess_text(row['UResponse'])
        original_response = preprocess_text(row['OResponse'])

        for algo_name, algo in algorithms.items():
            similarity_score = algo(response, original_response)
            df.at[index, algo_name] = similarity_score  # Ajouter le
score dans la colonne correspondante

    return df

def classify_similarity(similarity):
    if similarity >= 0.75:
        return 'cut'
    elif similarity >= 0.50:
        return 'heavy'
    elif similarity >= 0.25:
        return 'light'
    else:
        return 'non'

# Définir les algorithmes de similarité
algorithms = {
    'Jaro-Winkler': jaro_winkler_similarity,
    'N-GRAMM': ngram_similarity,
    'Set features': set_features_similarity,
    'Word Order Similarity': word_order_similarity,
}

# Appliquer la fonction sur votre DataFrame
df_syntaxic = apply_syntaxic_similarity(new_df, algorithms)
df_syntaxic
```

```
C:\Users\dscon\AppData\Local\Temp\ipykernel_9760\2836459194.py:46:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[algo_name] = 0.0  # Initialiser chaque colonne avec des scores à
0.0
C:\Users\dscon\AppData\Local\Temp\ipykernel_9760\2836459194.py:46:
```

```
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[algo_name] = 0.0  # Initialiser chaque colonne avec des scores à
0.0
C:\Users\dscon\AppData\Local\Temp\ipykernel_9760\2836459194.py:46:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[algo_name] = 0.0  # Initialiser chaque colonne avec des scores à
0.0
C:\Users\dscon\AppData\Local\Temp\ipykernel_9760\2836459194.py:46:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[algo_name] = 0.0  # Initialiser chaque colonne avec des scores à
0.0

    User Task                                              OResponse  \
0   g0pA    a  In object-oriented programming, inheritance is...
1   g0pB    a  In object-oriented programming, inheritance is...
2   g0pC    a  In object-oriented programming, inheritance is...
3   g0pD    a  In object-oriented programming, inheritance is...
4   g0pE    a  In object-oriented programming, inheritance is...
..   ...  ...                                                ...
90  g3pC    e  In mathematics and computer science, dynamic p...
91  g4pB    e  In mathematics and computer science, dynamic p...
92  g4pC    e  In mathematics and computer science, dynamic p...
93  g4pD    e  In mathematics and computer science, dynamic p...
94  g4pE    e  In mathematics and computer science, dynamic p...

                                          UResponse Category  Jaro-
Winkler  \
0   Inheritance is a basic concept of Object-Orien...      non
0.040767
1   Inheritance is a basic concept in object orien...      non
0.026549
2   inheritance in object oriented programming is ...    heavy
```

```
0.020779
3   Inheritance in object oriented programming is ...      cut
0.132450
4   In object-oriented programming, inheritance is...    light
0.966419
..                                                 ...      ...
...
90  In computer science and mathematics, dynamic p...    light
0.110737
91  In mathematics and computer science, dynamic p...      cut
0.615753
92  In mathematics and computer science, dynamic p...    light
0.043729
93  Dynamic programming is a method of providing s...    heavy
0.044361
94   Dynamic programming is a method for efficient...      non
0.030061

     N-GRAMM  Set features  Word Order Similarity
0   0.010453      0.080645               0.175141
1   0.006431      0.103627               0.129944
2   0.066929      0.191860               0.310734
3   0.373737      0.533333               0.559322
4   0.897143      0.909091               0.920904
..       ...           ...                    ...
90  0.126394      0.238372               0.185053
91  0.540741      0.591954               0.679715
92  0.400749      0.549708               0.459075
93  0.171975      0.326425               0.377224
94  0.030387      0.140426               0.206406

[95 rows x 9 columns]

# classer les similarités
df_syntaxic['Category_JARO'] = df_syntaxic['Jaro-
Winkler'].apply(classify_similarity)
df_syntaxic['Category_NGRAM'] = df_syntaxic['N-
GRAMM'].apply(classify_similarity)
df_syntaxic['Category_Set'] = df_syntaxic['Set
features'].apply(classify_similarity)
df_syntaxic['Category_WordOrder'] = df_syntaxic['Word Order
Similarity'].apply(classify_similarity)
df_syntaxic.head()

   User Task                                           OResponse  \
0  g0pA    a  In object-oriented programming, inheritance is...
1  g0pB    a  In object-oriented programming, inheritance is...
2  g0pC    a  In object-oriented programming, inheritance is...
3  g0pD    a  In object-oriented programming, inheritance is...
4  g0pE    a  In object-oriented programming, inheritance is...
```

```
                                            UResponse Category  Jaro-
Winkler  \
0  Inheritance is a basic concept of Object-Orien...      non
0.040767
1  Inheritance is a basic concept in object orien...      non
0.026549
2  inheritance in object oriented programming is ...    heavy
0.020779
3  Inheritance in object oriented programming is ...      cut
0.132450
4  In object-oriented programming, inheritance is...    light
0.966419

    N-GRAMM  Set features  Word Order Similarity Category_JARO
Category_NGRAM  \
0  0.010453      0.080645               0.175141           non
non
1  0.006431      0.103627               0.129944           non
non
2  0.066929      0.191860               0.310734           non
non
3  0.373737      0.533333               0.559322           non
cut
4  0.897143      0.909091               0.920904         heavy
heavy

  Category_Set Category_WordOrder
0          non                non
1          non                non
2          non                cut
3        light              light
4        heavy              heavy

# Accuracy
print("Jaro-Winkler Accuracy:", sum(df_syntaxic['Category_JARO'] ==
df_syntaxic['Category'])/ len(new_df) *100)
print("N-GRAMM Accuracy:", sum(df_syntaxic['Category_NGRAM'] ==
df_syntaxic['Category'])/ len(new_df) * 100)
print("Set features Accuracy:", sum(df_syntaxic['Category_Set'] ==
df_syntaxic['Category'])/ len(new_df) * 100)
print("Word Order Similarity Accuracy:",
sum(df_syntaxic['Category_WordOrder'] == df_syntaxic['Category'])/
len(new_df) * 100)

Jaro-Winkler Accuracy: 44.21052631578947
N-GRAMM Accuracy: 48.421052631578945
Set features Accuracy: 53.68421052631579
Word Order Similarity Accuracy: 51.578947368421055
```

Accuracy

Jaro-Winkler Accuracy: 44.21052631578947

N-GRAMM Accuracy: 48.421052631578945

Set features Accuracy: 53.68421052631579

Word Order Similarity Accuracy: 51.578947368421055

```
from sklearn.metrics import jaccard_score
import numpy as np
import pandas as pd

# Calculate n-grams
def ngrams(text, n=3):
    words = text.split()
    return set([' '.join(words[i:i+n]) for i in range(len(words)-
n+1)])

# Calculate the Jaccard index
def jaccard_index(doc1, doc2, n=3):
    ngrams_doc1 = ngrams(doc1, n)
    ngrams_doc2 = ngrams(doc2, n)
    intersection = ngrams_doc1.intersection(ngrams_doc2)
    union = ngrams_doc1.union(ngrams_doc2)
    return len(intersection) / len(union) if len(union) > 0 else 0  #
Avoid division by zero

# Classification based on the similarity score
def classify_similarity(similarity_score):
    if similarity_score < 0.2:
        return 'non'
    elif 0.2 <= similarity_score < 0.5:
        return 'cut'
    elif 0.5 <= similarity_score < 0.75:
        return 'light'
    else:
        return 'heavy'

# Compute semantic similarity for the DataFrame
def compute_semantic_similarity(df):
    semantic_results = []

    for i in range(len(df)):
        doc1 = df['OResponse'][i]
        doc2 = df['UResponse'][i]

        semantic = SemanticDistanceDocs(doc1, doc2)
        similarity_class = classify_similarity(semantic)
```

```
        semantic_results.append({
            'Semantic_Similarity': semantic,
            'Semantic_Category': similarity_class
        })

    semantic_df = pd.DataFrame(semantic_results)
    df = pd.concat([df.reset_index(drop=True), semantic_df], axis=1)
    return df

result = compute_semantic_similarity(new_df)

new_df.to_csv('final_df.csv', index=False)

new_df.columns

Index(['User', 'Task', 'OResponse', 'UResponse', 'Category', 'Jaro-
Winkler',
       'N-GRAMM', 'Set features', 'Word Order Similarity',
'Category_JARO',
       'Category_NGRAM', 'Category_Set', 'Category_WordOrder'],
      dtype='object')

result.columns

Index(['User', 'Task', 'OResponse', 'UResponse', 'Category', 'Jaro-
Winkler',
       'N-GRAMM', 'Set features', 'Word Order Similarity',
'Category_JARO',
       'Category_NGRAM', 'Category_Set', 'Category_WordOrder',
       'Semantic_Similarity', 'Semantic_Category'],
      dtype='object')

new_df.head()

   User Task                                           OResponse  \
0  g0pA    a  In object-oriented programming, inheritance is...
1  g0pB    a  In object-oriented programming, inheritance is...
2  g0pC    a  In object-oriented programming, inheritance is...
3  g0pD    a  In object-oriented programming, inheritance is...
4  g0pE    a  In object-oriented programming, inheritance is...

                                          UResponse Category  Jaro-
Winkler  \
0  Inheritance is a basic concept of Object-Orien...      non
0.040767
1  Inheritance is a basic concept in object orien...      non
0.026549
2  inheritance in object oriented programming is ...    heavy
0.020779
3  Inheritance in object oriented programming is ...      cut
0.132450
```

```
4  In object-oriented programming, inheritance is...    light
0.966419

    N-GRAMM  Set features  Word Order Similarity Category_JARO
Category_NGRAM  \
0  0.010453      0.080645                0.175141           non
non
1  0.006431      0.103627                0.129944           non
non
2  0.066929      0.191860                0.310734           non
non
3  0.373737      0.533333                0.559322           non
light
4  0.897143      0.909091                0.920904           cut
cut

  Category_Set Category_WordOrder
0          non                non
1          non                non
2          non              light
3        heavy              heavy
4          cut                cut
```

```
print(f'Accuracy of Semantic Similarity: {sum(new_df["Category"] ==
result["Semantic_Category"])/len(new_df) * 100}%')
```

```
Accuracy of Semantic Similarity: 20.0%
```

Accuracy of Semantic Similarity: 20.0%