MASTER 2 DATA SCIENCE ET SOCIÉTÉ NUMÉRIQUE

MACHINE LEARNING ET ÉPIDÉMIOLOGIE SOCIALE

RAPPORT FINAL

Professeur : Adrien UGON

Dong Pha PHAM
Matthieu COMOY
June Camille MENARD

Table des matières

1. Introduction

Les réseaux sociaux sont devenus un nouveau moyen offrant à tous une arène pour le partage des points de vue et perspectives sur différents problèmes et sujets. Les pensées, états d'esprit, p
ositions sur des problèmes sociaux et politiques spécifiques à travers des textes, des photos, des messages audio/vidéo et autres types de publications. En effet, malgré la disponibilité d'autres formes de communication, le texte reste l'un des moyens de communication les plus courants sur Twitter qui sera a été choisi dans le cadre de notre travail qui consistera à détecter et à analyser à la fois les sentiments et les émotions exprimés par les personnes à travers des textes dans leurs publications Twitter. Des tweets ont été collectés et un ensemble de données a été créé avec des informations sur le texte, l'utilisateur, les émotions et les sentiments.

La reconnaissance des émotions est le processus d'identification des émotions humaines en s'appuyant simplement sur les compétences personnelles et l'interprétation de l'interaction interpersonnelle, en automatisant le processus avec un système informatique ou par une approche semi-automatisée combinant les deux. Reconnaître avec précision les émotions humaines est une tâche laborieuse en raison de la polyvalence et de l'ambiguïté des émotions. Les mêmes émotions peuvent être exprimées de plusieurs manières, et plusieurs émotions ont parfois la même expression. Les émotions peuvent varier en fonction de la personnalité, du sexe, du lieu, de l'origine ethnique, de la culture, de la situation, en plus de nombreux autres paramètres psychologiques, sociaux et individuels. Parfois, détecter l'émotion réelle d'une personne à partir d'un morceau de texte, de son discours ou de ses expressions faciales est difficile, même pour un autre humain. Lorsqu'il s'agit de détection automatique des émotions par un ordinateur, le niveau de complication du problème est est encore plus élevé.
L'objectif de cette recherche est la détection et l'analyse des émotions et des sentiments à partir du texte de Twitter.
L'analyse de textes pour y détecter la présence d'états affectifs, leur polarité, les émotions associées et les opinions exprimées a suscité de nombreux travaux ces dernières années. Cet intérêt toujours croissant peut s'expliquer, en partie, par les perspectives d'applications qu'elle ouvre dans de nombreux domaines tels que : les systèmes de recommandation en ligne , l'intelligence économique, la veille politique et gouvernementale,...etc. Les méthodes appliquées sont généralement spécifiques aux types de textes traités. La majorité des méthodes proposées ont été créées pour l'Anglais et pour la polarité, avec quelques travaux récents qui existent aussi pour le Français et pour les émotions. Dans ce travail, nous nous intéresserons à la classification des tweets en langue française.

2. Méthodologie proposée

Pour commencer, la première tâche à laquelle nous serons confrontés est de scrapper des tweets. Nous allons cibler des tweets de personnes politiques Françaises ayant beaucoup de visibilité à l'aide de twint. Pour cibler ces personnalités nous allons scraper une liste web trouvée sur internet à l'aide de beautifulsoup.
Pour la seconde tâche qui consiste à analyser les sentiments des tweets que nous avons scrappés nous allons utiliser la librairie python NLTK qui est très utilisée pour l'analyse de sentiment. Pour cela il faut avant nettoyer le texte des tweets qui contiennent souvent du bruit comme des liens ou des emojis. Au final nous aurons des scores sur plusieurs catégories de sentiments (les classes ont été considérées telles que : Joy (Joie), fear (peur), sadness (tristesse), angry (colère), surprise (surprise), disgust (dégoût).

La dernière tâche à laquelle nous sommes confrontés sera de sélectionner un sentiment principal par tweet et de le faire correspondre à un sentiment dans le fichier de l'ontologie des sentiments que nous allons manipuler avec Owlready2 et pandas. Nous allons cibler ces sentiments par leur IRI que nous allons enfin ajouter à chaque tweet correspondant pour obtenir un dataframe complet.

3. Théorie

Dans ce travail, après avoir extrait des tweets sur divers sujets, ils ont été prétraités et divisés en mots, puis les parties de discours correspondantes ont été identifiées pour ces mots. Après avoir analysé les mots, ils ont été classés en émotions. Les émotions ont ensuite été analysées et l'analyse a révélé des résultats intéressants et de nouveaux défis.

4. Nos travaux

A) Description de corpus

Le corpus choisi est une compilation de tweets des 50 personnes politiques les plus influentes sur twitter en France. Nous avons choisi ce sujet pour constituer notre corpus de tweets car nous pensons qu'il est intéressant d'analyser le langage qu'utilisent les différents politiques. Et comme tout sujet politique , il vaut mieux rester le plus neutre possible pour l'analyse. Aussi nous avons basé la sélection des personnages politiques sur une liste trouvée ….

B) Scrapping des tweets sur Twitter

Twint est une librairie python permettant de scraper des comptes ou des mots-clés sur Twitter. Plusieurs outils font déjà ce travail, alors quel est l'intérêt de cette librairie ? En deux points :

- Utilisation sans inscription au préalable
- Nous ne passons pas par l'API Twitter. Il n'y a donc aucune limite dans la récupération du nombre de tweets !

Le github de Twint : https://github.com/twintproject/twint

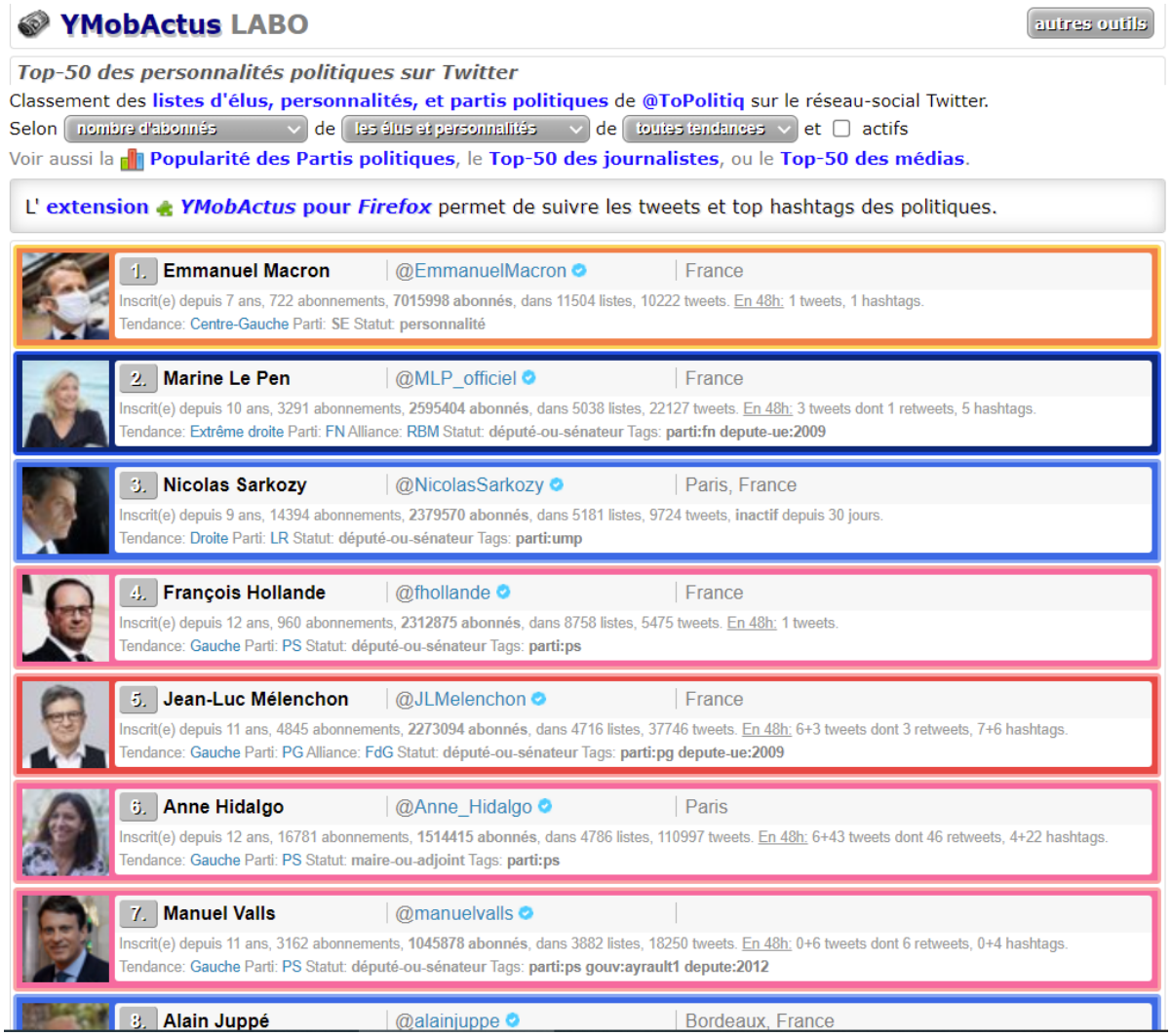
L'avantage de Twint est que vous n'avez pas besoin de l'API de Twitter pour faire fonctionner TWINT. Twint utilise les opérateurs de recherche de Twitter pour vous permettre de :
- gratter les Tweets d'utilisateurs spécifiques
- gratter les Tweets relatifs à certains sujets
- hashtags et tendances
- ou trier les informations sensibles des Tweets comme les e-mails et les numéros de téléphone.

En guise de présentation, nous allons nous amuser à récupérer l'ensemble des tweets des politiques les plus médiatisés de notre cher pays.

Installation de Twint: Nous installons la librairie via pip

```
!pip install tWint
```

Nous avons commencé par chercher une page internet recensant les comptes Twitter des politiques. Nous sommes tombés sur ce site :
http://ymobactus.miaouw.net/labo-top-politiques.php?mode=followers&liste=personnalites&tendance.



Pour scrapper les tweets avec twint, il faut d'avoir leur identifiant. Pour scraper tous les identifiants des comptes Twitter sur ce site, nous utiliserons la librairie BeautifulSoup. La bibliothèque python BeautifulSoup permet d'extraire des informations d'un site web, ou encore d'un document XML, avec quelques lignes de code.

```
!pip install beautifulsoup4
import requests
from bs4 import BeautifulSoup
```

En examinant le code html de la page, nous nous rendons compte que l'info que nous cherchons se situe dans la balise <a> :

Voici le code permettant de récupérer leur identifiant

```
# Déclarer de la liste des comptes
compte_twitter = []
stopwords = []

# Faire la demande "GET" à une URL
r = requests.get('http://ymobactus.miaouw.net/labo-top-politiques.php?mode=followers&liste=personnalites&tendance')

# Extraire le contenu
d = r.content

# Créer un objet soupe
soup = BeautifulSoup(d)

# Récupérer du "href" contenant le lien url redirigeant vers le compte Twitter de chaque personnalité
handles = [ a["href"] for a in soup.find_all("a", href=True) if("twitter" in a["href"])]

# Nous bouclons afin de ne récupérer que la fin de url (nom du compte)
#Pour d'avoir uniquement le nom du compte, nous utilison REGEX

for url in handles:
    compte = re.search('[^/]+(?=/$|$)',url)
#Ajouter des noms à la liste compte_twitter
    compte_twitter.append(compte.group())
```

Et voici ce que nous récupérons en sortie :

```
['lists', 'topolitiq', 'EmmanuelMacron', 'MLP_officiel', 'NicolasSarkozy', 'fhollande', 'JLMelenchon', 'Anne_Hidalgo', 'manuelv
alls', 'alainjuppe', 'najatvb', 'EPhilippe_LH', 'benoithamon', 'ChTaubira', 'nk_m', 'RoyalSegolene', 'Lagarde', 'bayrou', 'fleu
rpellerin', 'FrancoisFillon', 'BrunoLeMaire', 'montebourg', 'CecileDuflot', 'MarionMarechal', 'vpecresse', 'CCastaner', 'lauren
twauquiez', 'jf_cope', 'olivierveran', 'JeanCASTEX', 'jeanmarcayrault', 'LaurentFabius', 'GDarmanin', 'BGriveaux', 'GilbertColl
ard', 'dupontaignan', 'f_philippot', 'nadine__morano', 'MichelBarnier', 'BCazeneuve', 'pierremoscovici', 'JLBorloo', 'xavierber
trand', 'cestrosi', 'edgarmorinparis', 'axellelemaire', 'ECiotti', 'aurelifil', 'jpraffarin', 'lepenjm', 'MartineAubry', 'datir
achida', 'ymobactus', 'ymobactus']
```

Nous allons choisir de scrapper des tweets de Monsieur Jean-Luc Mélenchon avec son identifiant de Twitter : "**JLMelenchon**". Nous allons utiliser Twint pour récupérer ses tweets.

Nous importons les bibliographies. Si vous souhaitez utiliser Twint via un Jupyter Notebook, il est important d'installer et d'importer la bibliothèque nest_asyncio.

```
import twint
import requests
import pandas as pd
import nest_asyncio
import numpy as np
```

Nous allons faire un programme pour faire du tweet scraping en interrogeant le compte de "**JLMechenlon**".

```
# Configure
c = twint.Config()
c.Username = "JLMelenchon"
c.Limit = 10
c.Store_csv = True
c.Output = "tweets.csv"
# Run
nest_asyncio.apply()
twint.run.Search(c)
```

Le programme n'interroge que les tweets sur le compte "JLMechenlon" et récupère jusqu'à 10 tweets récents.

Alors, quel est le but du programme ci-dessus, nous allons expliquer un par un, comme suit:

- c.Username = ici vous remplissez la requête que vous souhaitez rechercher
- Limite = Limiter le nombre de tweets qui sont grattés
- c.Store_csv : Activer pour enregistrer les tweets dans le fichier CSV.
- c.Output : enregistrer les données de sortie par nom ou répertoire spécifique

Et nous récupérons en sortie un dataset “tweets.csv”

C) Analyse de sentiment des tweets représentant les émotions détectées

Dans les quelques lignes au-dessus, nous apporterons les bibliothèques et le module nécessaires qui vont nous aider à faire le traitement des données à partir de la bibliothèque nltk.
Nous stockons simplement les stopword français, la fonction stemmer dans différentes variables dans les lignes suivantes.

```
import pandas as pd
import io
import requests
url="http://advanse.lirmm.fr/FEEL.csv"
s=requests.get(url).content
c=pd.read_csv(io.StringIO(s.decode('utf-8')),  sep = ';')
```

```
import re, string, unicodedata
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
```

- Le package requests permet d’effectuer programmatiquement des requêtes HTTP (comme celles que fait un navigateur).
- io est un module qui permet de gérer des flux d'entrées et de sorties
- nltk : Supprimer les stopword en Python (NLTK - Natural Language Toolkit en python a une liste de mots vides stockés dans 16 langues différentes)

Le prétraitement des données est la CLÉ

Dans les projets NLP, la partie la plus importante est le prétraitement des données, afin qu'elles soient prêtes à être utilisées par un modèle. C'est ce que nous faisons maintenant.
Comme il y a beaucoup de paramètres inutiles dans l'ensemble de données, extrayons et stockons l'entrée et la sortie, qui est la colonne tweets. La prochaine étape est une étape de pré-traitement des données.

Les tweets collectés contiennent en général du bruit (par exemple, des symboles supplémentaires et manquants, des caractères, des fautes d'orthographe). L'ensemble de données a été prétraité en supprimant les parties inutiles afin d'avoir plus de convivialité. Tout d'abord, @username (par exemple, @mary), les URL (par exemple, http://a.com/) et les émoticônes (par exemple, ':-)') ont été supprimés de l'ensemble de données. Ensuite, plusieurs caractères (par exemple, happyyyyy) ont également été supprimés. Les symboles de hashtag ont été supprimés sans supprimer les mots qui leur sont associés. Après avoir supprimé ces symboles, la redondance de l'ensemble de données a été vérifiée. Enfin, l'ensemble de données filtré contenait des tweets sans émoticônes, hashtags, occurrences multiples inutiles d'un symbole ou de mots illisibles.

Pourquoi et que devons-nous supprimer des stopword ?

Les stopword sont les mots du langage naturel qui ont très peu de sens, tels que : “ un, une, des, le, la, les, ...etc. Les moteurs de recherche et autres plates-formes d'indexation d'entreprise filtrent souvent les mots vides tout en récupérant les résultats de la base de données par rapport aux requêtes des utilisateurs.
Les mots vides sont souvent supprimés du texte avant la formation des modèles d'apprentissage en profondeur et d'apprentissage automatique, car les mots vides sont produits en abondance, fournissant ainsi peu ou pas d'informations uniques pouvant être utilisées pour la classification ou le regroupement.
Pour supprimer les mots inutiles, je vais utiliser les techniques suivantes :
Supprimer les mots vides : essentiellement des mots comme celui-ci, un, un, des, le, la,les,... etc qui n'affecte pas le sens du tweet. Suppression de la ponctuation : ',.* !' et d'autres signes de ponctuation qui ne sont pas vraiment nécessaires pour le modèle.

Voici les fonctions pour les supprimer:

```
def remove_URL(sample):
    """Remove URLs from a sample string"""
    return re.sub(r"https\S+", "", sample)

def remove_stopword(words):
    return [word for word in words.split() if word not in stopwords.words('french')]

def preprocess(txt):
    txt = txt.lower()
    txt = remove_URL(txt)
    txt = re.sub(r'\W', ' ', txt) # Remove all the special characters
    txt = "".join(filter(lambda x: not x.isdigit(), txt)) #remove number
    txt = re.sub('[!@#$]', '', txt)

    token_after_removing_stopword = remove_stopword(txt)

    txt = ' '.join(token_after_removing_stopword)

    return txt
```

Nous importons le dataset et nous choisirons la colonne “tweets” pour analyser les tweets. Nous importons le fichier “FEEL.csv” pour analyser le sentiment des tweets plus tard.
FEEL : Lexique des émotions en français, il est un lexique français contenant plus de 14 000 mots distincts exprimant des émotions et des sentiments. Il suit les deux polarités de base et six émotions d'Ekman (Ekman, 1992). Il a été créé en traduisant et en développant automatiquement le lexique émotionnel anglais NRC-Canada (Mohammad & Turney, 2013).

```
df = pd.read_csv('tweets.csv', encoding='utf8')
feel = pd.read_csv('C:/Users/phamd/Downloads/FEEL.csv', encoding='utf8', delimiter=';')

df['tweet'].head()
```

Voici le résultat:

```
0    Rendez-vous à 12h pour #QuestionsPol sur Franc...
1    Il faut réquisitionner les usines permettant d...
2    La semaine de l'élection législative qui décoi...
3    🚤 Ce dimanche 6 juin, si vous habitez le 20e a...
4    «L'Europe qui protège», cette imposture au seu...
Name: tweet, dtype: object
```

La lemmatisation est par définition une action consistant à l’analyse lexicale d’un texte avec pour but de regrouper les mots d’une même famille. On parle ici de donner la forme canonique d’un mot ou d’un ensemble de mots : Chacun de ces mots d’un contenu donné se trouve réduit en une entité appelée en lexicologie lemme ou encore “forme canonique d’un mot”. Les lemmes d’une langue utilisent plusieurs formes en fonction :

du genre (masculin ou féminin),
de leur nombre (un ou plusieurs),
de leur personne (moi, toi, eux…),
de leur mode (indicatif, impératif…)

De plus, parfois, le même mot peut avoir plusieurs « lemmes » différentes. Ainsi, en fonction du contexte dans lequel il est utilisé, nous identifions donc la balise “partie du discours” (POS) pour le mot dans ce contexte spécifique et extraire la lemme appropriée. Des exemples de mise en œuvre de ceci sont fournis dans les sections suivantes.

Pour travailler sur les lemmes et les étiquettes morphosyntaxiques, nous utilisons la TreeTagger, avec la librairie Python “treetaggerwrapper”, permettant de connecter Python à TreeTagger**. (***TreeTagger est un tagueur et lemmatiseur de point de vente très rapide ayant des performances très acceptables sur tous les langages TermSuite*)

```
# TREETAGGER LEMMATIZER
import pandas as pd
import treetaggerwrapper as tt

t_tagger = tt.TreeTagger(TAGLANG ='fr', TAGDIR ='C:\TreeTagger')
df['clean_tweet'] = df['tweet'].apply(lambda txt: preprocess(txt))
df['pos_tags'] = df['clean_tweet'].apply(lambda txt: t_tagger.tag_text(txt))
```

- TAGLANG (string) – code de langue pour les textes (‘en’,’fr’,…), nous choisissons la langue français "fr"
- TAGDIR (string) – chemin d'accès au répertoire d'installation de TreeTagger

Nous traitons chaque ligne par colonne de tweet en l'enregistrant en tant que colonne "clean_tweet". Nous appliquons le résultat des fonctions de suppression des stopwords ci-dessus et puis nous créons une colonne "pos_tags", après le traitement et l'étiquette.

Voici le résultat :

```
0    [rendez\tVER:pres\trendre, h\tABR\theure, ques...
1    [faut\tVER:pres\tfalloir, réquisitionner\tVER:...
2    [semaine\tNOM\tsemaine, élection\tNOM\télectio...
3    [dimanche\tNOM\tdimanche, juin\tNOM\tjuin, si\...
4    [europe\tADJ\teurope, protège\tVER:pres\tproté...
Name: pos_tags, dtype: object
```

D) Lemmatisation et POS

Un baliseur de parties du discours POS a été utilisé pour marquer chaque mot des tweets dans les parties du discours correspondantes. Types- Conjonction de coordination, Nombre cardinal, Déterminant, Mot étranger, Préposition ou conjonction de subordination, Adjectif, Marqueur d'élément de liste, Modal, Nom - singulier, Nom - pluriel, Nom propre - singulier, Nom propre - pluriel, Prédéterminant, Terminaison possessive, Pronom personnel, Pronom possessif, Pronom comparatif, Adverbe, Symbole, Interjection, Verbe - forme de base, Verbe - participe présent, Verbe - participe passé, Verbe-Conditionnel.
Nous divisons le lemme et la pos en colonnes, ou utilisons d'abord une boucle “for” pour parcourir chaque ligne de la colonne "pos_tags", puis utilisons à nouveau la boucle continue pour parcourir chaque mot d'un tweet (c'est-à-dire mot par mot sur une ligne de colonne "pos_tags"). On tronque la chaîne à chaque tabulation, le premier élément enregistre le mot d'origine, le deuxième élément est pour stocker la balise gérondif, le dernier élément est pour stocker le lemme et puis nous sauvegardons le résultat dans un dataframe.

```
original = []
lemmas = []
tags = []

sentences_pos_tags = df['pos_tags'].tolist()

res = pd.DataFrame(columns=['original', 'lemma', 'tag'])

for pos_tags in sentences_pos_tags:
    original = []
    lemmas = []
    tags = []
    for t in pos_tags:
        original.append(t.split('\t')[0])
        tags.append(t.split('\t')[1])
        lemmas.append(t.split('\t')[-1])
    res = res.append({'original': original, 'lemma': lemmas, 'tag': tags}, ignore_index=True)
res
```

Voici le résultat :

	original	lemma	tag
0	[rendez, h, questionspol, france, inter, franc...	[rendre, heure, questionspol, france, inter, f...	[VER:pres, ABR, NOM, NOM, NOM, NOM, NOM, ADJ]
1	[faut, réquisitionner, usines, permettant, pro...	[falloir, réquisitionner, usine, permettre, pr...	[VER:pres, VER:infi, NOM, VER:ppre, VER:infi, ...
2	[semaine, élection, législative, décoiffe, dim...	[semaine, élection, législatif, décoiffer, dim...	[NOM, NOM, ADJ, VER:pres, NOM, VER:pres, VER:i...
3	[dimanche, juin, si, habitez, e, arrondissemen...	[dimanche, juin, si, habiter, e, arrondissemen...	[NOM, NOM, KON, VER:pres, VER:pper, NOM, NOM, ...
4	[europe, protège, cette, imposture, seul, serv...	[europe, protéger, ce, imposture, seul, servic...	[ADJ, VER:pres, PRO:DEM, NOM, ADJ, NOM, NOM, V...
...	...	...	...
95	[hostile, régime, place, biélorussie, livré, a...	[hostile, régime, placer, biélorussie, livrer,...	[ADJ, NOM, VER:pres, NOM, VER:pper, VER:pres, ...
96	[fiche, vaccin, russe, autre, nationalité, int...	[ficher, vaccin, russe, autre, nationalité, in...	[VER:pres, NOM, ADJ, ADJ, NOM, VER:pres, ADJ, ...
97	[ça, a, problème, voir, tous, dirigeants, gauc...	[cela, avoir, problème, voir, tout, dirigeant,...	[PRO:DEM, VER:pres, NOM, VER:infi, PRO:IND, NO...
98	[pass, sanitaire, injuste, puisqu, sait, possi...	[pass, sanitaire, injuste, puisqu, savoir, pos...	[NOM, ADJ, ADJ, NOM, VER:pres, ADJ, ADV, VER:i...
99	[défendu, licence, libre, vaccins, dès, début,...	[défendre, licence, libre, vaccin, dès, début,...	[VER:pper, NOM, ADJ, NOM, PRP, NOM, ADJ, VER:p...

100 rows × 3 columns

Nous obtenons un dataframe avec 100 lignes.

E) Lexique des émotions

L'analyse des sentiments permet l'évaluation sémantique d'un texte en fonction des sentiments et opinions exprimés. Alors qu'une grande attention a été portée à la polarité (positive, négative) des mots anglais, seules quelques études se sont intéressées aux émotions véhiculées (joie, colère, surprise, tristesse,...etc.) surtout dans d'autres langues. Nous importons un nouveau lexique français (FEEL) considérant à la fois la polarité et l'émotion (liens : http://advanse.lirmm.fr/feel.php).

Pour l'application à notre projet, nous lisons le dataset "FEEL.csv" et nous convertissons l'étiquette "positive" en 1 et l'étiquette "negative" en -1, le but est de créer un vecteur à 7 dimensions pour le traitement des sentiments plus tard. Le premier élément représente la polarité "Polarity", et le six éléments sont les émotions : joie, colère, tristesse, dégoût, surprise et peur (joy, fear, sadness, angry, surprise, disgust). Ensuite, nous les convertissons en numpy.array. Nous prenons la première colonne du fichier "FEEL.csv" que nous mettons dans le dataframe Words et la deuxième colonne du fichier "FEEL.csv" que nous mettons dans le dataframe "emotions" puis nous les combinons pour former un dictionnaire.

	id	word	polarity	joy	fear	sadness	anger	surprise	disgust
0	1	à ce endroit là	positive	0	0	0	0	0	0
1	2	à le hâte	negative	0	1	0	0	1	0
2	3	à part	negative	0	0	1	0	0	0
3	4	à pic	negative	0	1	0	0	0	0
4	5	à rallonge	negative	0	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...
14122	14125	zozoter	negative	0	1	1	1	0	0
14123	14126	merci	positive	1	0	0	0	0	0
14124	14127	remercier	positive	1	0	0	0	0	0
14125	14128	remerciment	positive	1	0	0	0	0	0
14126	14129	moins	negative	0	0	0	0	0	0

14127 rows × 9 columns

```
# load feel.csv
feel = pd.read_csv('C:/Users/phamd/Downloads/FEEL.csv', encoding='utf8', delimiter=';')
feel['polarity'] = feel['polarity'].replace('positive', 1)
feel['polarity'] = feel['polarity'].replace('negative', -1)
words = feel[feel.columns[1]].tolist()
emotions = feel[feel.columns[2:]].to_numpy()

dict_emotion = dict(zip(words, emotions))
dict_emotion
```

Voici le résultat:

```
{'à ce endroit là': array([1, 0, 0, 0, 0, 0, 0], dtype=int64),
 'à le hâte': array([-1,  0,  1,  0,  0,  1,  0], dtype=int64),
 'à part': array([-1,  0,  0,  1,  0,  0,  0], dtype=int64),
 'à pic': array([-1,  0,  1,  0,  0,  0,  0], dtype=int64),
 'à rallonge': array([-1,  0,  0,  1,  0,  0,  0], dtype=int64),
 'abasourdir': array([-1,  0,  0,  0,  0,  1,  0], dtype=int64),
 'ablation': array([-1,  0,  1,  0,  0,  0,  1], dtype=int64),
 'abominable': array([-1,  0,  1,  0,  0,  0,  1], dtype=int64),
 'abrupt': array([-1,  0,  1,  0,  0,  0,  0], dtype=int64),
 'absent': array([-1,  0,  1,  1,  0,  0,  0], dtype=int64),
 'absorber': array([1, 0, 0, 0, 0, 1, 0], dtype=int64),
 'absurde': array([-1,  0,  0,  0,  1,  1,  1], dtype=int64),
 'acceptable': array([1, 0, 0, 0, 0, 0, 0], dtype=int64),
 'accidentel': array([-1,  0,  1,  1,  0,  1,  0], dtype=int64),
 'accusateur': array([-1,  0,  1,  0,  1,  0,  1], dtype=int64),
 'accuser': array([-1,  0,  1,  1,  1,  0,  1], dtype=int64),
 'acheteur': array([1, 0, 0, 0, 0, 0, 0], dtype=int64),
 'acquisition': array([1, 0, 0, 0, 0, 0, 0], dtype=int64),
 'active': array([1, 0, 0, 0, 0, 0, 0], dtype=int64),
 'actuel': array([1, 0, 0, 0, 0, 1, 0], dtype=int64)
```

Voici les résultats de 10 tweets dans la colonne "lemma" ci-dessus

```
lemmas = res['lemma'].apply(lambda x: ' '.join(x)).tolist() #get lemma
lemmas[0:10]
```

```
['rendre heure questionspol france inter france info tv',
 'falloir réquisitionner usine permettre produire france silicium soutien salarier ferropem clavaux',
 'semaine élection législatif décoiffer dimanche faire élire simonnet assemblée national simonnetdéputée',
 'dimanche juin si habiter e arrondissement pari connaître gens habiter voter faire voter simonnet devenir e députer insoumis',
 'europe protéger ce imposture seul service profit refuser licence libre vaccin covid',
 'valls hollande venir faire campagne ème contre simonnet tout monde comprendre écouter zombie voter danielle simonnet simonnet députée circo',
 'rendre dimanche juin heure france inter france info tv questionspol',
 'rdls ligne thème répression militant bure députer insoumis benedicttaurine violenter castex ficher compteur linky devenir pay ant arnaque total',
 'samedi juin partout france marcher liberté contre idée extrême droit tout infos',
 'beaucoup monde soutien simonnet législatif partiel e arrondissement pari dimanche juin rendre urne faire élire ème députer in soumis']
```

À partir de la colonne “lemma”, nous convertissons en chaîne (string) pour créer une colonne contenant des tweets, chaque élément est une ligne de lemma (tweet), et puis nous analysons le sentiment pour chaque tweet. Nous traitons chacun des 100 tweets ci-dessus avec une boucle “for”:

```
# for loop in each tweet
for lemma in lemmas:
    tokens = [w for w in word_tokenize(lemma)] #tokenize tweet
    print('twwet', lemma)
    print('Token', tokens)
    break
```

Voici le résultat:

```
twwet rendre heure questionspol france inter france info tv
Token ['rendre', 'heure', 'questionspol', 'france', 'inter', 'france', 'info', 'tv']
```

Ici, worrl_tokenize(lemma) remplit des missions de tokenizer. Donc, c'est quoi un tokenizer?. Un tokenizer est un outil fondé sur un algorithme basé sur un ensemble de règles ou sur un apprentissage à partir d'un corpus étiqueté manuellement. Il permet de découper le texte en mots. C'est une analyse dite morphologique. La tokenisation est un travail de segmentation qui décompose une phrase en ses multiples éléments.
En Traitement Automatique des Langues (TAL), la tokenisation fait partie du processus de normalisation. Elle segmente le texte en entrée en unités linguistiques manipulables comme les mots, la ponctuation, les nombres, les données alphanumériques... Chaque élément correspond à un token qui sera utile à l'analyse.
Le but de ce procédé est de séparer les unités de base d'un texte qui se prêteront à une analyse pointue par la suite. On peut penser qu'il suffit de détecter les espaces entre les mots, mais ce n'est pas toujours aussi facile - en particulier pour le français.

Ensuite, nous allons utiliser la boucle for token in tokens pour analyser le sentiment de chaque mot, si leur token est dans la liste des mots du fichier FEEL.csv (voici le dictionnaire que j'ai créé ci-dessus), puis vérifier quelle est leur polarité dans le dictionnaire. Cela signifie que vous aurez un tableau de 7 éléments (c'est-à-dire 1 vecteur à 7 dimensions), le premier élément du tableau représente la polarité, les 6 éléments restants représentent les niveaux émotionnels. Dict_emotion[token][0] == 1 c'est-à-dire positif, dict_emotion[token][0] == -1 c'est-à-dire négatif, et ainsi de suite à chaque fois que nous comptons le nombre de positifs et de négatifs pour comparer le don et la polarité pour les tweets entiers.

- si positive est supérieur à négative alors le tweet est positif
- sinon, positive est inférieur à négative alors le tweet est négatif
- ou si positive et négative sont égaux alors le tweets est neutre

```
# polarity
if n_pos > n_neg:
    polarity.append('positive')
elif n_pos < n_neg:
    polarity.append('negative')
else:
    polarity.append('neutral')
```

Ensuite, nous calculons la moyenne de l'émotion, en créant 1 vecteur vide : vector = np.zeros(n_emotion) pour stocker la valeur de l'émotion, ce vecteur est 1x6, c'est-à-dire : dict_emotion[token][1 :], ici, array s'exécutera à partir de 1. Et l'index à la position 0 stocke la

polarité, et les valeurs de la position 1 à la fin stockent l'émotion : vector = np.add(vector,dict_emotion[token][1:], out=vector). Nous ajoutons ensuite chaque vecteur avec chaque token jusqu'à la fin de la chaîne de token *(le vecteur d'initialisation est 0)*.

Nous vérifierons si le token a des sentiments dans le fichier (FEEL.csv) puis nous les ajouterons.

```
    for token in tokens:
        if token in dict_emotion:
            # labeled polarity for tweet
            if dict_emotion[token][0] == 1:
                label='positive'
                n_pos += 1
            else:
                label = 'negative'
                n_neg += 1
            print("({0} | {1})".format(token, label), end=', ')

            # labeled emotion for tweet
            vector = np.add(vector,dict_emotion[token][1:], out=vector) # sum emotion vectors
            # print(dict_emotion[token][1:]) uncomment to debug
            count += 1
        else:
            print(token, end=', ')

    # emotion sentiment
    if count != 0:
```

```
            print(token, end=', ')

    # emotion sentiment
    if count != 0:
        vector /= count #normalize by mean
    for emotion_label, emotion_value in zip(emotion, vector):
        tmp_sentiment.update({emotion_label : float(emotion_value)}) # {'joy': 0.0, 'fear': 0.0, 'sadness': 0.0, 'angry': 0.16,

    sentiment.append(tmp_sentiment)

    # polarity
```

count!=0 est utilisé pour montrer que la chaîne de tokens contient un token d'émotion dedans. Et si chaîne contient un token d'émotion, nous ajoutons les vecteurs et divisons la moyenne.

```
    vector /= count #normalize by mean
for emotion_label, emotion_value in zip(emotion, vector):
    tmp_sentiment.update({emotion_label : float(emotion_value)}) # {'joy': 0.0, 'fear': 0.0, 'sadness': 0.0, 'angry': 0.16,
```

La boucle for ci-dessus est destinée à la gestion des sauvegardes structurées. Enfin, nous imprimons l'émotion et la polarité pour chaque tweet. Nous obtiendrons le résultat comme la photo ci-dessous:

```
    # print output for each tweet
    print()
    print("\n->Emotion: ", list(tmp_sentiment.items()))
    print("\n->Polarity: ", polarity[-1])
    print()
    print('>'*100)
    print()
```

```
(gourou | negative), (candidat | positive), (programmer | positive), (rassembler | positive),
(comprendre | positive), (gens | positive), venir, (droit | positive), (supporter | negative)
```

Ensuite, nous créons des colonnes pour stocker les résultats de polarité et des émotions.

	original	lemma	tag	polarity
0	[rendez, h, questionspol, france, inter, franc...	[rendre, heure, questionspol, france, inter, f...	[VER:pres, ABR, NOM, NOM, NOM, NOM, NOM, ADJ]	positive
1	[faut, réquisitionner, usines, permettant, pro...	[falloir, réquisitionner, usine, permettre, pr...	[VER:pres, VER:infi, NOM, VER:ppre, VER:infi, ...	positive
2	[semaine, élection, législative, décoiffe, dim...	[semaine, élection, législatif, décoiffer, dim...	[NOM, NOM, ADJ, VER:pres, NOM, VER:pres, VER:i...	positive
3	[dimanche, juin, si, habitez, e, arrondissemen...	[dimanche, juin, si, habiter, e, arrondissemen...	[NOM, NOM, KON, VER:pres, VER:pper, NOM, NOM, ...	positive
4	[europe, protège, cette, imposture, seul, serv...	[europe, protéger, ce, imposture, seul, servic...	[ADJ, VER:pres, PRO:DEM, NOM, ADJ, NOM, NOM, V...	positive
...	...	...	...	...
95	[hostile, régime, place, biélorussie, livré, a...	[hostile, régime, placer, biélorussie, livrer,...	[ADJ, NOM, VER:pres, NOM, VER:pper, VER:pres, ...	positive
96	[fiche, vaccin, russe, autre, nationalité, int...	[ficher, vaccin, russe, autre, nationalité, in...	[VER:pres, NOM, ADJ, ADJ, NOM, VER:pres, ADJ, ...	positive
97	[ça, a, problème, voir, tous, dirigeants, gauc...	[cela, avoir, problème, voir, tout, dirigeant,...	[PRO:DEM, VER:pres, NOM, VER:infi, PRO:IND, NO...	neutral
98	[pass, sanitaire, injuste, puisqu, sait, possi...	[pass, sanitaire, injuste, puisqu, savoir, pos...	[NOM, ADJ, ADJ, NOM, VER:pres, ADJ, ADV, VER:i...	positive
99	[défendu, licence, libre, vaccins, dès, début,...	[défendre, licence, libre, vaccin, dès, début,...	[VER:pper, NOM, ADJ, NOM, PRP, NOM, ADJ, VER:p...	positive

100 rows × 4 columns

	joy	fear	sadness	angry	surprise	disgust
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	0.000000	0.100000	0.200000	0.200000	0.300000	0.000000
4	0.222222	0.111111	0.111111	0.222222	0.000000	0.222222
...	...	...	...	...	...	...
95	0.000000	0.416667	0.083333	0.250000	0.083333	0.166667
96	0.142857	0.000000	0.000000	0.142857	0.000000	0.142857
97	0.000000	0.400000	0.200000	0.300000	0.200000	0.100000
98	0.000000	0.000000	0.111111	0.222222	0.111111	0.111111
99	0.076923	0.076923	0.076923	0.076923	0.000000	0.000000

100 rows × 6 columns

Ensuite, nous joignons les deux dataframes ensemble, nous obtenons le résultat comme suit:

	original	lemma	tag	polarity	joy	fear	sadness	angry	surprise	disgust
0	[rendez, h, questionspol, france, inter, franc...	[rendre, heure, questionspol, france, inter, f...	[VER:pres, ABR, NOM, NOM, NOM, NOM, NOM, ADJ]	positive	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	[faut, réquisitionner, usines, permettant, pro...	[falloir, réquisitionner, usine, permettre, pr...	[VER:pres, VER:infi, NOM, VER:ppre, VER:infi, ...	positive	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	[semaine, élection, législative, décoiffe, dim...	[semaine, élection, législatif, décoiffer, dim...	[NOM, NOM, ADJ, VER:pres, NOM, VER:pres, VER:i...	positive	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	[dimanche, juin, si, habitez, e, arrondissemen...	[dimanche, juin, si, habiter, e, arrondissemen...	[NOM, NOM, KON, VER:pres, VER:pper, NOM, NOM, ...	positive	0.000000	0.100000	0.200000	0.200000	0.300000	0.000000
4	[europe, protège, cette, imposture, seul, serv...	[europe, protéger, ce, imposture, seul, servic...	[ADJ, VER:pres, PRO:DEM, NOM, ADJ, NOM, NOM, V...	positive	0.222222	0.111111	0.111111	0.222222	0.000000	0.222222
...	...	...	...	...	...	...	...	...	...	...
95	[hostile, régime, place, biélorussie, livré, a...	[hostile, régime, placer, biélorussie, livrer,...	[ADJ, NOM, VER:pres, NOM, VER:pper, VER:pres, ...	positive	0.000000	0.416667	0.083333	0.250000	0.083333	0.166667
96	[fiche, vaccin, russe, autre, nationalité, int...	[ficher, vaccin, russe, autre, nationalité, in...	[VER:pres, NOM, ADJ, ADJ, NOM, VER:pres, ADJ, ...	positive	0.142857	0.000000	0.000000	0.142857	0.000000	0.142857
97	[ça, a, problème, voir, tous, dirigeants, gauc...	[cela, avoir, problème, voir, tout, dirigeant,...	[PRO:DEM, VER:pres, NOM, VER:infi, PRO:IND, NO...	neutral	0.000000	0.400000	0.200000	0.300000	0.200000	0.100000
98	[pass, sanitaire, injuste, puisqu, sait, possi...	[pass, sanitaire, injuste, puisqu, savoir, pos...	[NOM, ADJ, ADJ, NOM, VER:pres, ADJ, ADV, VER:i...	positive	0.000000	0.000000	0.111111	0.222222	0.111111	0.111111
99	[défendu, licence, libre, vaccins, dès, début,...	[défendre, licence, libre, vaccin, dès, début,...	[VER:pper, NOM, ADJ, NOM, PRP, NOM, ADJ, VER:p...	positive	0.076923	0.076923	0.076923	0.076923	0.000000	0.000000

100 rows × 10 columns

Nous avons divisé en 2 parties nos résultats, la polarité et l'émotion sont gérées séparément. La polarité que nous calculons est basée sur la majorité de la polarité de chaque jeton et nous ne considérons pas le facteur émotion, donc certains jetons ne montrent pas d'émotion mais ont une polarité telle que: ***acquisition;positif;0;0;0;0;0;0/ actif;positif;0;0;0;0; 0,0/ charnel;négatif;0;0;0;0;0;0.*** Donc les sentiments : joy, fear, sadness, anger, surprise, disgust ne déterminent pas complètement la polarité du tweet. En d'autres termes, la limitation de l'algorithme ici est que les émoticônes du tweet ne représentent pas entièrement la polarité du tweet, parce que certains mots positifs et négatifs n'ont pas d'expressions émotionnelles comme les 3 mots ci-dessus.

F) Ontology avec Owlready :

En informatique et en data science, une ontologie est l'ensemble structuré des termes et concepts représentant le sens d'un champ d'informations, que ce soit par les métadonnées d'un espace de noms, ou les éléments d'un domaine de connaissances. L'ontologie constitue en soi un modèle de données représentatif d'un ensemble de concepts dans un domaine, ainsi que des relations entre ces concepts. Elle est employée pour raisonner à propos des objets du domaine concerné. Ainsi, l'« ontologie est aux données ce que la grammaire est au langage ».

Le terme ontologie utilisé par analogie avec le concept philosophique d'ontologie qui est l'étude de l'être en tant qu'être, c'est-à-dire l'étude des propriétés générales de ce qui existe. Les concepts sont donc organisés dans un graphe dont les relations peuvent être, des relations sémantiques ; des relations de subsomption.

Cette partie se base sur le fichier d'ontologie mfoem.owl, L'ontologie des émotions, développé par trois chercheurs du centre des sciences affective de suiise et l'université de Buffalo. Il comprend plus de 613 classes qui définissent les comportements et émotions humaines sous forme d'ontologie, c'est-à-dire avec des relations hiérarchiques.

```
[5]  import owlready2
     from owlready2 import *
     onto_path.append("/content/Owlready")
     onto = owlready2.get_ontology("http://purl.obolibrary.org/obo/mfoem.owl")
     onto.load()
```

Une fois l'ontologie chargée, nous pouvons accéder dynamiquement à toutes ses classes. Nous avons commencé par chercher les classes correspondant à celles utilisées pour l'analyse de sentiment, c'est-à-dire joie, peur, tristesse, colère, surprise, dégoût.

```
[40] sentiments = ['joy', 'fear', 'sadness', 'anger', 'surprise', 'disgust']
     for sentiment in sentiments:
       print(sentiment+' : '+ str(onto.search(label = sentiment)))
```

```
joy : [obo.MFOEM_000034]
fear : [obo.MFOEM_000026]
sadness : [obo.MFOEM_000056]
anger : [obo.MFOEM_000009]
surprise : [obo.MFOEM_000032]
disgust : [obo.MFOEM_000019]
```

Pour finir nous avons ajouté les codes IRI des ontologies au dataframe des tweets en fonction de leur émotion principale trouvée avec l'analyse de sentiment. Le code IRI associé à chaque tweet permet d'identifier l'émotion principale détectée dans le texte du tweet.

```
[30] tweets['max_emotion'] = tweets[['joy','fear','sadness','angry','surprise','disgust']].idxmax(axis=1)
     tweets['IRI'] = tweets['max_emotion']

     tweets['IRI'] = tweets['IRI'].replace(to_replace ="joy", value ="obo.MFOEM_000034")
     tweets['IRI'] = tweets['IRI'].replace(to_replace ="fear", value ="obo.MFOEM_000026")
     tweets['IRI'] = tweets['IRI'].replace(to_replace ="sadness", value ="obo.MFOEM_000056")
     tweets['IRI'] = tweets['IRI'].replace(to_replace ="angry", value ="obo.MFOEM_000009")
     tweets['IRI'] = tweets['IRI'].replace(to_replace ="surprise", value ="obo.MFOEM_000032")
     tweets['IRI'] = tweets['IRI'].replace(to_replace ="disgust", value ="obo.MFOEM_000019")

     tweets.tail(5)
```

	original	lemma	tag	polarity	joy	fear	sadness	angry	surprise	disgust	max_emotion	IRI
95	['hostile', 'régime', 'place', 'biélorussie', ...	['hostile', 'régime', 'placer', 'biélorussie',...	['ADJ', 'NOM', 'VER:pres', 'NOM', 'VER:pper', ...	positive	0.000000	0.416667	0.083333	0.250000	0.083333	0.166667	fear	obo.MFOEM_000026
96	['fiche', 'vaccin', 'russe', 'autre', 'nationa...	['ficher', 'vaccin', 'russe', 'autre', 'nation...	['VER:pres', 'NOM', 'ADJ', 'ADJ', 'NOM', 'VER:...	positive	0.142857	0.000000	0.000000	0.142857	0.000000	0.142857	joy	obo.MFOEM_000034
97	['ça', 'a', 'problème', 'voir', 'tous', 'dirig...	['cela', 'avoir', 'problème', 'voir', 'tout', ...	['PRO:DEM', 'VER:pres', 'NOM', 'VER:infi', 'PR...	neutral	0.000000	0.400000	0.200000	0.300000	0.200000	0.100000	fear	obo.MFOEM_000026
98	['pass', 'sanitaire', 'injuste', 'puisqu', 'sa...	['pass', 'sanitaire', 'injuste', 'puisqu', 'sa...	['NOM', 'ADJ', 'ADJ', 'NOM', 'VER:pres', 'ADJ'...	positive	0.000000	0.000000	0.111111	0.222222	0.111111	0.111111	angry	obo.MFOEM_000009

5) Analyse des limites du programme et conclusion

- Pour le programme, la limitation de l'algorithme ici est que les émoticônes du tweet ne représentent pas entièrement la polarité du tweet, parce que certains mots positifs et négatifs n'ont pas d'expressions émotionnelles. Plus précisément, un token qui ne montre pas d'émotion mais a une polarité, par exemple:
l'acquisition;positif;0;0;0;0;0;0/ actif;positif;0;0;0;0;0;0/ charnel;négatif;0;0;0;0;0;0, parce que dans le dictionnaire ces tokens n'ont pas d'émotions, donc lors de l'ajout du vecteur

0 avec des token émotionnels, il change les poids. Donc ici surprise ou joie ou colère, ...etc ne déterminera pas complètement la polarité du tweet complet.

- Cette première limite se reflète également sur la labellisation ontologique des tweets. Comme le programme se base sur des sentiments prédéfinis nous ne pouvons pas prendre en compte tout le spectre des sentiments qui est présent dans le fichier de l'ontologie des sentiments.

6) Réferences

- Emotion and Sentiment Analysis from Twitter Text - Sailunaz, Kashfia (https://prism.ucalgary.ca/handle/1880/107533)
- Owlready2 Documentation - Jean-Baptiste LAMY (https://readthedocs.org/projects/owlready2/downloads/pdf/stable/)