Project: Council of the European Union

Group 2 : Yingqing Chen, Pourtaud Robin DS

Step 2: Definition of our approach

1 - Natural Language processing tool to use

For this project, we will use Python3.11 and the following packages:

- Pandas
- Numpy
- Plotly
- Sklearn
- PDFminer
- hdbscan

And the following natural language processing package:

- Spacy

2 - DM/ML methods:

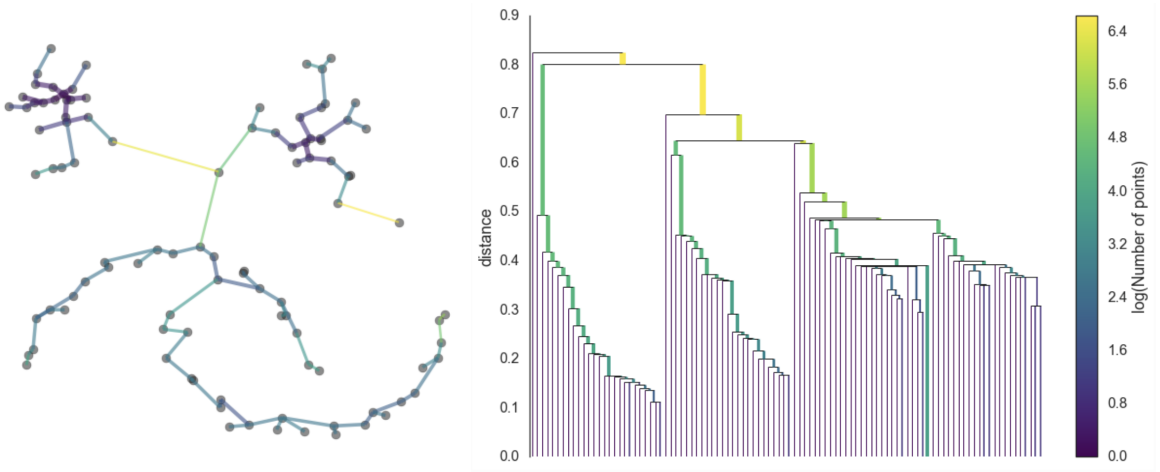
After the preprocessing of the document (and after projecting the words into an embedding with doc2vec or fastext or….) , we will perform multiples ML techniques:

- SVM will be our Supervised Classification method.
- HDBSCAN will be our Unsupervised Clustering method

We will use the followings packages to use SVM and HDBSCAN:

- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- https://github.com/scikit-learn-contrib/hdbscan

Therefore, we will be able to deduce similar words and group them. HDBSCAN has multiple benefits in comparison to a simple cluster technique like K-means. HDBSCAN is density based, therefore the cluster is not limited in terms of shapes. Furthermore, HDBSCAN can produce a hierarchy of clusters in a dendrogram. The followings plot describe its potential:



We can see on the dendrogram that there are 4 main clusters. The two moons are however quite close, but still distinct.

We hope that using HDBSCAN makes sense in our problem. It is possible that similar words are more likely to get a ball shape, therefore K-means can be sufficient. We will see.

3 - Lexical units/words/terms to extract

Comment: This was our proposition to do, but we talked with Maxime and agreed with his step for building the ontology (work with textblob and then lemmatize). The following is our first proposition.

For ontology learning, we first need to transform the PDF into text. We will use the package PDFminer to do so.

Therefore, with Spacy, we can start by extracting all sentences.
Considering these sentences independently, we will find different lexical units with different targets:

1. **Nouns:** Words that represent a person, place, thing, or idea.
2. **Verbs:** Words that express action or state of being.
3. **Adjectives:** Words that describe or modify nouns or pronouns.
4. **Adverbs:** Words that describe or modify verbs, adjectives, or other adverbs.
5. **Name entities:** Words that replace nouns or noun phrases. Proper Nouns.
6. **Prepositions:** Words that indicate relationships between nouns or pronouns and other words in a sentence.

It can be express in our ontology has:

1. **Nouns:** Representing classes or concepts.
2. **Verbs:** Representing relations or properties between classes or concepts.
3. **Adjectives:** Representing attributes of classes or concepts.
4. **Adverbs:** Representing qualifiers or modifiers of relations or properties.
5. **Name:** Representing instances or individuals of classes or concepts.
6. **Prepositions:** Representing the relationships between instances or individuals of classes or concepts in the ontology.

4 - Feature Selection

Obviously, we don't want to consider the whole corpus. We want to keep only words and sentences that respect some metric and pattern constraints:

- **Sentences that are related to core concepts:** One approach could be to use keyword matching or text similarity methods to identify sentences that contain key terms related to our core concept. Some verbs, or words like "as", "like" etc can give us more information.

- **Core concepts co-occurrences:** Identifying the co-occurrences of core concepts in the corpus can help to identify patterns of association between concepts and to extract important relationships.

Step 3 and 4:

To avoid redundancy, the answers to step 3 and 4 are sent by Maxime.

Not being sure if Maxime sent you this part, we joined the OWL file.

The OWL file is **not rebalanced** but is however joined to the document. We saw your message about not sending the OWL file, but since we have one in work, here it is :

It was generated by the following code:

```
import pandas as pd
df = pd.read_csv("https://docs.google.com/spreadsheets/d/16OlZEn2__3ALyECxYS00vlkECWGORTsfKs3ygWjaue8/export?gid=0&format=csv")
def addIndividual(currentClass, superClass):
    return """
    <owl:Class rdf:about="http://www.semanticweb.org/AI-law#{}">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/AI-law#{}"/>
    </owl:Class>
    """.format(currentClass, superClass)
rdfIndividuals = ""
for i, row in df.iterrows():
    if not str(row["Core concept"]) == "nan":
        rdfIndividuals+= addIndividual(str(row["noun_phrase"]).replace(" ", "_"), str(row["Core concept"]).replace(" ", "_"))
with open("onto.rdf", "w") as f:
    f.write(rdfIndividuals)
```

(the rdf file is then appended at the end of the core ontology).