Validation du projet IA :

Groupe 9 :

- Malak Ben Salem
- Hela Darguechi
- Khalil Ennaifer
- Oumayma Talbi
- Mohamed Amorri
- Hanine Djebbi

Phase 1:

Extraction et nettoyage des données : nous avons utilisé les méthodes de base de Python et les expressions régulières pour extraire nos données de notre PDF primaire PMBOK et les nettoyer pour obtenir des données plus compréhensibles.
-- Méthodes NLP : pour obtenir des données plus propres et de meilleure qualité, nous utilisons ensuite des techniques NLP bien connues telles que :
La suppression des mots d'arrêt : vise à supprimer certains mots d'arrêt du texte, des mots qui n'ont pas de pertinence significative et qui peuvent être supprimés des documents.
La tokenisation : consiste à décomposer une séquence de caractères en morceaux (mots/phrases) appelés tokens. Balisage des parties du discours : balises pour chaque mot (si le mot est un nom, un verbe, un adjectif, etc.).
Lemmatisation : Processus consistant à trouver le lemme d'un mot en fonction de son sens et de son contexte. Vise à supprimer les terminaisons flexionnelles.

Les bibliothèques:

Les expressions régulières: sont utilisées dans pratiquement tous les langages. C'est un outil qui permet de vérifier si le contenu d'une variable a la forme de ce que l'on attend. Ils permettent aussi de modifier ou de supprimer tous les éléments indésirables dans une variable.
Pdfplumber : est une bibliothèque Python pour l'extraction de texte et de table.

Entrée [1]:

```
import pdfplumber
import numpy as np
import pandas as pd
import re
import string
from re import search

import spacy
from spacy.matcher import Matcher
from spacy.tokens import span
from spacy import displacy

import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import WordNetLemmatizer,SnowballStemmer
from nltk.corpus import wordnet, stopwords
from nltk import pos_tag, RegexpParser
import sys
```

Prétraitement et analyse des données à partir de textes non structurés

Chapitre Scope , Schedule et Cost:

EXTRACTION DES SOUS_TITRES ET LEURS CONTENUS:

Entrée [2]:

```
def process_segmentation(separateur) :
    name = ''
    start = 0
    process_df = pd.DataFrame(columns=['Process Name','start'])
    with pdfplumber.open("PMBOK.pdf") as pdf:
        for i in range(152,269) :
            page = (pdf.pages[i]).extract_text()
            for line in page.split('\n') :
                if re.match('\d.\d\s[A-Z]+\s[A-Z]', line ) :
                    if start != 0 :
                        process_df = process_df.append({'Process Name': name,'start':start}

                    name = line[0:]
                    start = i

        process_df = process_df.append({'Process Name': name,'start':start} , ignore_index=
        return (process_df)
```

Entrée [3]:

```
d1 = process_segmentation('.')
d1
```

Out[3]:

	Process Name	start
0	5.1 PLAN SCOPE MANAGEMENT	156
1	5.2 COLLECT REQUIREMENTS	159
2	5.3 DEFINE SCOPE	170
3	5.4 CREATE WBS	175
4	5.5 VALIDATE SCOPE	181
5	5.6 CONTROL SCOPE	184
6	6.1 PLAN SCHEDULE MANAGEMENT	191
7	6.2 DEFINE ACTIVITIES	195
8	6.3 SEQUENCE ACTIVITIES	199
9	6.4 ESTIMATE ACTIVITY DURATIONS	205

EXTRACTION DES SOUS_SOUS TITRES ET LEURS CONTENUS:

Entrée [4]:

```
def process_segmentation2(separateur) :
    name = ''
    start = 0
    process_df = pd.DataFrame(columns=['Process Name','start'])
    with pdfplumber.open("PMBOK.pdf") as pdf:
        for i in range(152,269) :
            page = (pdf.pages[i]).extract_text()
            for line in page.split('\n') :
                if re.match('\d.\d.\d\s[A-Z]+\s[A-Z]', line ) :
                    if start != 0 :
                        process_df = process_df.append({'Process Name': name,'start':start}

                    name = line[0:]
                    start = i

        process_df = process_df.append({'Process Name': name,'start':start} , ignore_index=
        return (process_df)
```

Entrée [5]:

```
d2 = process_segmentation2('.')
```

EXTRACTION DES SOUS_SOUS_SOUS TITRES ET LEURS CONTENUS:

Entrée [6]:

```
def process_segmentation3(separateur) :
    ch=''
    name = ''
    start = 0
    process_df = pd.DataFrame(columns=['Process Name','start'])
    with pdfplumber.open("PMBOK.pdf") as pdf:
        for i in range(152,269) :
            page = (pdf.pages[i]).extract_text()
            for line in page.splitlines():
                if re.match('\d.\d.\d.\d\s[A-Z]+\s[A-Z]',line) or re.match('\d.\d.\d.\d\s[A
                    if start != 0 :
                        process_df = process_df.append({'Process Name': name,'start':start}

                    name = line[0:]
                    ch = ''
                    start = i

                ch = ch + separateur +line

        process_df = process_df.append({'Process Name': name,'start':start} , ignore_index=
        return (process_df)
```

Entrée [7]:

```
d3 = process_segmentation3('.')
```

Entrée [8]:

```
d3
```

Out[8]:

	Process Name	start
0	5.1.1.1 PROJECT CHARTER	156
1	5.1.1.2 PROJECT MANAGEMENT PLAN	156
2	5.1.1.3 ENTERPRISE ENVIRONMENTAL FACTORS	157
3	5.1.1.4 ORGANIZATIONAL PROCESS ASSETS	157
4	5.1.2.1 EXPERT JUDGMENT	157
...	...	...
191	7.4.3.1 WORK PERFORMANCE INFORMATION	264
192	7.4.3.2 COST FORECASTS	264
193	7.4.3.3 CHANGE REQUESTS	264
194	7.4.3.4 PROJECT MANAGEMENT PLAN UPDATES	264
195	7.4.3.5 PROJECT DOCUMENTS UPDATES	266

196 rows × 2 columns

Entrée [9]:

```
Sommaire = []
Sommaire = pd.concat([d1,d2,d3])

Sommaire = Sommaire.sort_values(by='Process Name', ignore_index=True)

Sommaire.drop('start', inplace=True, axis=1)
Sommaire.reset_index(drop=True, inplace=True)
Sommaire.insert(1,column='Contenu',value='')

#*********************************************************************************************

Sommaire.loc[-1] = ["PROJECT SCOPE MANAGEMENT",""]
Sommaire = Sommaire.sort_index().reset_index(drop=True)

Sommaire.loc[89.90] = ["PROJECT SCHEDULE MANAGEMENT",""]
Sommaire = Sommaire.sort_index().reset_index(drop=True)

Sommaire.loc[194.195] = ["PROJECT COST MANAGEMENT",""]
Sommaire = Sommaire.sort_index().reset_index(drop=True)
Sommaire
```

Out[9]:

	Process Name	Contenu
0	PROJECT SCOPE MANAGEMENT	
1	5.1 PLAN SCOPE MANAGEMENT	
2	5.1.1 PLAN SCOPE MANAGEMENT: INPUTS	
3	5.1.1.1 PROJECT CHARTER	
4	5.1.1.2 PROJECT MANAGEMENT PLAN	
...	...	...
258	7.4.3.1 WORK PERFORMANCE INFORMATION	
259	7.4.3.2 COST FORECASTS	
260	7.4.3.3 CHANGE REQUESTS	
261	7.4.3.4 PROJECT MANAGEMENT PLAN UPDATES	

Entrée [10]:

```
ch = ''
with pdfplumber.open("PMBOK.pdf") as pdf:
    for i in range(152,269) :
        ch = ch + (pdf.pages[i]).extract_text()
print(ch)
```

```
5

PROJECT SCOPE MANAGEMENT
Project Scope Management includes the processes required to ensure that th
e project includes
all the work required, and only the work required, to complete the project
successfully. Managing the
project scope is primarily concerned with defining and controlling what is
and is not included in the
project.
The Project Scope Management processes are:
5.1  Plan Scope Management—The process of creating a scope management plan
that
documents how the project and product scope will be defined, validated, an
d controlled.
5.2  Collect Requirements—The process of determining, documenting, and man
aging
stakeholder needs and requirements to meet project objectives.
5.3 Define Scope—The process of developing a detailed description of the p
```

Entrée [11]:

```
for i in range (len(Sommaire)) :
    try :
        Corpus = ch[ch.index(Sommaire['Process Name'][i]) : ch.index((Sommaire['Process Nam
        Sommaire['Contenu'][i] = Corpus

    except :
        Corpus = ch[ch.index(Sommaire['Process Name'][i]) : ]
        Sommaire['Contenu'][i] = Corpus
Sommaire
```

Out[11]:

	Process Name	Contenu
0	PROJECT SCOPE MANAGEMENT	PROJECT SCOPE MANAGEMENT \nProject Scope Manag...
1	5.1 PLAN SCOPE MANAGEMENT	5.1 PLAN SCOPE MANAGEMENT \nPlan Scope Managem...
2	5.1.1 PLAN SCOPE MANAGEMENT: INPUTS	5.1.1 PLAN SCOPE MANAGEMENT: INPUTS \n \n
3	5.1.1.1 PROJECT CHARTER	5.1.1.1 PROJECT CHARTER \nDescribed in Section...
4	5.1.1.2 PROJECT MANAGEMENT PLAN	5.1.1.2 PROJECT MANAGEMENT PLAN \n \nDescribed...
...	...	...
258	7.4.3.1 WORK PERFORMANCE INFORMATION	7.4.3.1 WORK PERFORMANCE INFORMATION \nDescrib...

Entrée [12]:

```
for i in range(len(Sommaire)):
    if Sommaire['Contenu'][i].startswith(Sommaire['Process Name'][i]):
        Sommaire['Contenu'][i]=Sommaire['Contenu'][i][ len (Sommaire['Process Name'][i]) :
Sommaire
```

Out[12]:

	Process Name	Contenu
0	PROJECT SCOPE MANAGEMENT	\nProject Scope Management includes the proce...
1	5.1 PLAN SCOPE MANAGEMENT	\nPlan Scope Management is the process of crea...
2	5.1.1 PLAN SCOPE MANAGEMENT: INPUTS	\n \n
3	5.1.1.1 PROJECT CHARTER	\nDescribed in Section 4.1.3.1. The project ch...
4	5.1.1.2 PROJECT MANAGEMENT PLAN	\n \nDescribed in Section 4.2.3.1. Project man...
...	...	...
258	7.4.3.1 WORK PERFORMANCE INFORMATION	\nDescribed in Section 4.5.1.3. Work performan...
259	7.4.3.2 COST FORECASTS	\nEither a calculated EAC value or a bottom-up...
260	7.4.3.3 CHANGE REQUESTS	\nDescribed in Section 4.3.3.4. Analysis of pr...
261	7.4.3.4 PROJECT MANAGEMENT PLAN UPDATES	\nAny change to the project management plan go...
262	7.4.3.5 PROJECT DOCUMENTS UPDATES	\nProject documents that may be updated as a r...

263 rows × 2 columns

Entrée [13]:

```
#Add Column Type For Inputs , Outpus , TandT
Sommaire.insert(1,column='Type' , value='' )
```

Entrée [14]:

```
for i in range(len(Sommaire)):
    if 'INPUTS' in Sommaire['Process Name'][i]:
        Sommaire['Type'][i] = 'has_input'
    elif 'OUTPUTS' in Sommaire['Process Name'][i]:
        Sommaire['Type'][i] = 'has_output'
    elif 'TOOLS AND TECHNIQUES' in Sommaire['Process Name'][i]:
        Sommaire['Type'][i] = 'has_tools_and_techniques'
    else:
        Sommaire['Type'][i] = ''
    print(Sommaire['Process Name'][i] ,'==>' , Sommaire['Type'][i])
Sommaire
```

```
PROJECT SCOPE MANAGEMENT ==>
5.1 PLAN SCOPE MANAGEMENT  ==>
5.1.1 PLAN SCOPE MANAGEMENT: INPUTS  ==> has_input
5.1.1.1 PROJECT CHARTER  ==>
5.1.1.2 PROJECT MANAGEMENT PLAN  ==>
5.1.1.3 ENTERPRISE ENVIRONMENTAL FACTORS  ==>
5.1.1.4 ORGANIZATIONAL PROCESS ASSETS  ==>
5.1.2 PLAN SCOPE MANAGEMENT: TOOLS AND TECHNIQUES  ==> has_tools_and_techn
iques
5.1.2.1 EXPERT JUDGMENT  ==>
5.1.2.2 DATA ANALYSIS  ==>
5.1.2.3 MEETINGS  ==>
5.1.3 PLAN SCOPE MANAGEMENT: OUTPUTS  ==> has_output
5.1.3.1 SCOPE MANAGEMENT PLAN  ==>
5.1.3.2 REQUIREMENTS MANAGEMENT PLAN  ==>
5.2 COLLECT REQUIREMENTS  ==>
5.2.1 COLLECT REQUIREMENTS: INPUTS  ==> has_input
5.2.1.1 PROJECT CHARTER  ==>
5.2.1.2 PROJECT MANAGEMENT PLAN  ==>
```

Reference

Entrée [15]:

```
#Add column reference
Sommaire.insert(3,column='Reference',value='')
```

Entrée [16]:

```
ch1 =''
for i in range(len(Sommaire)):
    ch1 = Sommaire['Contenu'][i]

    if 'Described in' in Sommaire['Contenu'][i]:
        a = re.findall(r"Described in Section+ \d{1,2}.\d.\d.\d", ch1)
        for b in a:
            #print(b)
            Sommaire['Reference'][i] =  b

    elif 'depicted in' in Sommaire['Contenu'][i]:
        a = re.findall(r"depicted in Figure+ \d+-+\d{1,2}", ch1)
        for b in a:
            #print(b)
            Sommaire['Reference'][i] =  b
    elif 'shown in' in Sommaire['Contenu'][i]:
        a = re.findall(r"shown in Figure+ \d+-+\d{1,2}", ch1)
        for b in a:
            #print(b)
            Sommaire['Reference'][i] =  b
    elif 'described in' in Sommaire['Contenu'][i] :
        a = re.findall(r"described in Section+ \d{1,2}.\d.\d.\d", ch1)
        for b in a:
            #print(b)
            Sommaire['Reference'][i] =  b
    else:
        Sommaire['Reference'][i] = ''
```

Data Cleaning

Entrée [17]:

```
for i in range (len(Sommaire)):

    Sommaire['Contenu'][i] = re.sub("\n","", Sommaire['Contenu'][i])
    # Remove Repeated Characters
    Sommaire['Contenu'][i] = re.sub("(.)\\1{2,}", "\\1", Sommaire['Contenu'][i])
    #Remove uu
    pattern = r'uu'
    Sommaire['Contenu'][i] = re.sub(pattern, '', Sommaire['Contenu'][i])
    #Remove whitespace  from both sides of a string:
    Sommaire['Contenu'][i] = Sommaire['Contenu'][i].strip();
    # Remove unnecessary white spaces in between words
    Sommaire['Contenu'][i] = re.sub(' +', ' ', Sommaire['Contenu'][i])
    # Remove Non-English characters
    Sommaire['Contenu'][i] = re.sub(r'[^\x00-\x7f]',r'',  Sommaire['Contenu'][i])
    Sommaire['Contenu'][i] = re.sub(r'[*\d]', '', Sommaire['Contenu'][i])


Sommaire
```

Out[17]:

	Process Name	Type	Contenu	Reference
0	PROJECT SCOPE MANAGEMENT		Project Scope Management includes the processe...	
1	5.1 PLAN SCOPE MANAGEMENT		Plan Scope Management is the process of creati...	depicted in Figure 5-2
2	5.1.1 PLAN SCOPE MANAGEMENT: INPUTS	has_input		
3	5.1.1.1 PROJECT CHARTER		Described in Section .... The project charter ...	Described in Section 4.1.3.1
4	5.1.1.2 PROJECT MANAGEMENT PLAN		Described in Section .... Project management p...	Described in Section 8.1.3.1
...	...	...	...	...
258	7.4.3.1 WORK PERFORMANCE INFORMATION		Described in Section .... Work performance inf...	Described in Section 4.5.1.3

Entrée [18]:

```
# Removing punctuations in string without (. / ?)
for i in range (len(Sommaire)):
    punc = '''()-[]{};:'"\,<>/?@#$%^&*_~'''
    # Using loop + punctuation string
    for j in  Sommaire['Contenu'][i]:
        if j in punc:
            Sommaire['Contenu'][i] =  Sommaire['Contenu'][i].replace(j, "")
```

Entrée [19]:

```
for i in range (len(Sommaire)):
    # Remove punctuation
    Sommaire['Process Name'][i] = Sommaire['Process Name'][i].translate(str.maketrans('', '
    # Remove numbers
    Sommaire['Process Name'][i] = re.sub(r'\d+', '', Sommaire['Process Name'][i])
    #Remove whitespace  from both sides of a string:
    Sommaire['Process Name'][i] = Sommaire['Process Name'][i].strip();
```

Entrée [20]:

```
for i in range (len(Sommaire)):
    Sommaire['Contenu'][i]=Sommaire['Contenu'][i].replace("Part 1  Guide","")
    Sommaire['Contenu'][i] = Sommaire['Contenu'][i].replace('——>', '')
```

Definition

Entrée [21]:

```
#Add Column Definition
Sommaire.insert(4,column='Definition' , value='' )
```

Entrée [22]:

```
ch1 =''
for i in range(len(Sommaire)):
    ch1 = Sommaire['Contenu'][i]

    if 'Described in' in Sommaire['Contenu'][i]:
        Sommaire['Definition'][i] =  ch1 [  : ch1.index('Described in') ]

    elif 'described in' in Sommaire['Contenu'][i] :
        Sommaire['Definition'][i] =  ch1 [  : ch1.index('described in') ]

    elif 'depicted in' in Sommaire['Contenu'][i]:
        Sommaire['Definition'][i] =  ch1 [  : ch1.index('depicted in') ]

    elif 'shown in' in Sommaire['Contenu'][i]:
        Sommaire['Definition'][i] =  ch1 [  : ch1.index('shown in') ]

    else:
        Sommaire['Definition'][i] = ''
```

Entrée [23]:

```
Sommaire
```

Out[23]:

	Process Name	Type	Contenu	Reference	Definition
0	PROJECT SCOPE MANAGEMENT		Project Scope Management includes the processe...		
1	PLAN SCOPE MANAGEMENT		Plan Scope Management is the process of creati...	depicted in Figure 5-2	Plan Scope Management is the process of creati...
2	PLAN SCOPE MANAGEMENT INPUTS	has_input			
3	PROJECT CHARTER		Described in Section .... The project charter ...	Described in Section 4.1.3.1	
4	PROJECT MANAGEMENT PLAN		Described in Section .... Project management p...	Described in Section 8.1.3.1	
...	...	...	...	...	...
258	WORK PERFORMANCE INFORMATION		Described in Section .... Work performance inf...	Described in Section 4.5.1.3	
259	COST FORECASTS		Either a calculated EAC value or a bottomup EA...		
260	CHANGE REQUESTS		Described in Section .... Analysis of project ...	Described in Section 4.3.3.4	
261	PROJECT MANAGEMENT PLAN UPDATES		Any change to the project management plan goes...	Described in Section 7.3.3.1	Any change to the project management plan goes...
262	PROJECT DOCUMENTS UPDATES		Project documents that may be updated as a res...	Described in Section 11.2.3.1	Project documents that may be updated as a res...

263 rows × 5 columns

Entrée [24]:

```
# Delete all number in all text in dataset
import re
def clean_numbers(text):
    # tk=word_tokenize()
    text_tokens = word_tokenize(text)
    # remove numbers
    text_nonum = re.sub(r'\d+', '', text)
    return text_nonum
```

Entrée [25]:

```
Sommaire['Contenu']=Sommaire['Contenu'].apply(lambda text:clean_numbers(text))
```

Remove Stop words

Entrée [26]:

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

```
[nltk_data] Downloading package punkt to C:\Users\ben salem
[nltk_data]     malak\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[26]:

```
True
```

Entrée [27]:

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to C:\Users\ben salem
[nltk_data]     malak\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[27]:

```
True
```

Entrée [28]:

```
#remove stopword in english text
all_stopwords = stopwords.words('english')
sw_list = ["you're", "you've", "she's", "it's", 'who', 'whom', 'am',  "is", 'are', 'was', '
    'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing',"a",
    'other', 'some',  'no', 'nor', 'not', 'own', 'same',  's', 't', 'can', 'don', "don't",
    "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'cou
    'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", "haven't", 'i
    'will','such','as', 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't",'can',"it'
    'being', 'have', 'has','an', 'had', 'having', 'do', 'does', 'did', 'doing',"a", 'other'
     'now','will','such','as','can']
all_stopwords.extend(sw_list)

for i in range(len(Sommaire)):
    text_tokens = word_tokenize(Sommaire['Contenu'][i])
    tokens_filtered= [word for word in text_tokens if not word in all_stopwords]
    Sommaire['Contenu'][i] = ' '.join(tokens_filtered)
```

Stemming

Entrée [29]:

```
#Stemming
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

ps = PorterStemmer()

for i in range(len(Sommaire['Contenu'])):
# choose some words to be stemmed
    words = [Sommaire['Contenu'][i]]
    for w in words:
        Sommaire['Contenu'][i] = ps.stem(w)
```

Entrée [30]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
stemmer = nltk.stem.porter.PorterStemmer()
remove_punctuation_map = dict((ord(char), None) for char in string.punctuation)
```

Normalize

Entrée [31]:

```
def stem_tokens (text):
    return [stemmer.stem(token) for token in text]
```

Entrée [32]:

```
def normalize(text):
    return stem_tokens(nltk.word_tokenize(text.lower().translate(remove_punctuation_map)))
```

Similarity with Spacy

Entrée [33]:

```
# spacy.cli.download("en_core_web_md")
```

Entrée [34]:

```
nlp = spacy.load("en_core_web_md")
```

Entrée [35]:

```
def jaccard_similarity(x,y):
  """ returns the jaccard similarity between two lists """
  intersection_cardinality = len(set.intersection(*[set(x), set(y)]))
  union_cardinality = len(set.union(*[set(x), set(y)]))
  return intersection_cardinality/float(union_cardinality)

#jaccard_similarity(sentences[0], sentences[1])
```

Svnonvm

Entrée [36]:

```
#Add Column Synonym
Sommaire.insert(5,column='Synonym' , value='' )
```

Entrée [37]:

```
c = []
for token1 in nlp(Sommaire['Process Name'][1].lower()):
    for token2 in nlp(Sommaire['Contenu'][1]):
        if (jaccard_similarity(token1.text, token2.text)> 0.7) and (token1.similarity(token
                A = token1.text, token2.text, jaccard_similarity(token1.text, token2.text)
                c.append([token1.text, token2.text])
print(c)
```

```
[['plan', 'plans'], ['scope', 'process'], ['scope', 'process'], ['scope', 'p
rocess'], ['scope', 'process'], ['scope', 'process'], ['scope', 'process'],
['management', 'managed']]
```

Entrée [38]:

```
#Synonym
B=[]
for i in range (len(Sommaire)):
    for token1 in nlp(Sommaire['Process Name'][i].lower()):
          for token2 in nlp(Sommaire['Contenu'][i]):
            if (jaccard_similarity(token1.text, token2.text)> 0.7) and (token1.similarity(t
                A = token1.text, token2.text, jaccard_similarity(token1.text, token2.text)
                if [token1.text, token2.text] not in B:
                    B.append([token1.text, token2.text])
    Sommaire['Synonym'][i]= B
    B= []
```

```
C:\Users\BENSAL~1\AppData\Local\Temp/ipykernel_6464/1257799477.py:6: UserWar
ning: [W008] Evaluating Token.similarity based on empty vectors.
  if (jaccard_similarity(token1.text, token2.text)> 0.7) and (token1.similar
ity(token2) < 1):
```

Entrée [39]:

```
l=''
for i in range(len(Sommaire)):
    text = Sommaire['Contenu'][i]
    for j in range ( len(text)-1 ):
        if (text[j+1]!=text[j]) or (text[j+1]!='.') :
            l=l+text[j]
    Sommaire['Contenu'][i] = l
    l=''
```

Entrée [40]:

```
#CREATE PARTICULAR GRAMMER WITH CHUNKING
# import nltk

# def prepareForNLP(text):
#     sentences = nltk.sent_tokenize(text)
#     sentences = [nltk.word_tokenize(sent) for sent in sentences]
#     sentences = [nltk.pos_tag(sent) for sent in sentences]
#     return sentences

# def chunk(sentence):
#     chunkToExtract = """
#     VP: {<VBG.*>|<VBD.*>|<VBP.*>|<VBP.*><VBN.*>|<VB>|<VBZ.*><VBN.*>}
#     NP:{<NN.*>+ | <]NNS.*>+| <]JJ.*><NN.*>+}
#     CLAUSE: {<NP><VP><NP>}"""
#     parser = nltk.RegexpParser(chunkToExtract)
#     result = parser.parse(sentence)
#     for subtree in result.subtrees():
#         if subtree.label() == 'CLAUSE':
#             t = subtree
#             print(t)

# sentences = prepareForNLP(Sommaire['Contenu'][1])
# for sentence in sentences:
#     chunk(sentence)
```

Entrée [41]:

```
for i in range(len(Sommaire)):
    Sommaire['Contenu'][i]= Sommaire['Contenu'][i].replace("described section ", "")
```

DataFrame Auxiliaire

Entrée [42]:

```
# Create custom grammar rule to label occurrences of any number of nouns, followed by a ver
my_grammar = r"""
NOUNS_VERB_NOUN: {<N.*>+<V.*><N.*>}"""
from nltk import RegexpParser
# Function to create parse tree using custom grammar rules and PoS tagged text
def get_parse_tree(grammar, pos_tagged_text):
    cp = RegexpParser(grammar)
    parse_tree = cp.parse(pos_tagged_text)
    #parse_tree.draw()  # Visualise parse tree
    return parse_tree
```

Entrée [43]:

```
def get_labels_from_grammar(grammar):
    labels = []
    for line in grammar.splitlines()[1:]:
        labels.append(line.split(":")[0])
    return labels
```

Entrée [44]:

```
def get_phrases_using_custom_labels(parse_tree, custom_labels_to_get):
    matching_phrases = []
    for node in parse_tree.subtrees(filter=lambda x: any(x.label() == custom_l for custom_l
        # Get phrases only, drop PoS tags
        matching_phrases.append([leaf[0] for leaf in node.leaves()])
    return matching_phrases
```

Extract Subject

Entrée [45]:

```
my_grammar = r"""
NOUNS_VERB_NOUN: {<NN.*>+<V.*>+<JJ.*>*}"""
Sommaire['Subject1'] =""
for i in range(len(Sommaire)):
    if len(Sommaire['Contenu'][i]) != 0:
        text=Sommaire['Contenu'][i]
        tokens = nltk.word_tokenize(text)
        tags = nltk.pos_tag(tokens)
        text_parse_tree = get_parse_tree(my_grammar, tags)
        my_labels = get_labels_from_grammar(my_grammar)
        phrases = get_phrases_using_custom_labels(text_parse_tree, my_labels)
        for phrase in phrases:
            sentence =""
            for k in range(len(phrase)):
                sentence = sentence + phrase[k]+' '

            Sommaire['Subject1'][i] = Sommaire['Subject1'][i] + sentence + ';'

Sommaire
```

4	PROJECT MANAGEMENT PLAN		. project management plan components include l...	Described in Section 8.1.3.1		[[management, managed]]	project management plan components include lim...
...	...	...	...	...	...	...	...
258	WORK PERFORMANCE INFORMATION		. work performance information includes inform...	Described in Section 4.5.1.3		[]	work performance information includes ;informa...
259	COST FORECASTS		either calculated eac value bottomup eac value...			[]	value documented communicated ;
260	CHANGE REQUESTS		. analysis project performance may result	Described in Section 4.3.3.4		[[requests, result], [requests, request]]	request cost schedule baselines components

Entrée [46]:

```
my_grammar = r"""
NOUNS_VERB_NOUN: {<NN.*>+}"""
Sommaire['subject'] =""
for i in range(len(Sommaire)):
    if len(Sommaire['Subject1'][i]) != 0:
        text=Sommaire['Subject1'][i]
        tokens = nltk.word_tokenize(text)
        tags = nltk.pos_tag(tokens)
        text_parse_tree = get_parse_tree(my_grammar, tags)
        my_labels = get_labels_from_grammar(my_grammar)
        phrases = get_phrases_using_custom_labels(text_parse_tree, my_labels)
        for phrase in phrases:
            sentence =""
            for k in range(len(phrase)):
                sentence = sentence + phrase[k]+' '

            Sommaire['subject'][i] = Sommaire['subject'][i] + sentence + ';'
Sommaire
```

Out[46]:

	Process Name	Type	Contenu	Reference	Definition	Synonym	Subject1	su
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	project scope management includes ;project inc...	project manag ;project ;p
1	PLAN SCOPE MANAGEMENT		plan scope management process creating scope m...	depicted in Figure 5-2	Plan Scope Management is the process of creati...	[[plan, plans], [scope, process], [management,...	plan scope ;management process creating ;produ...	;manag pr ;p ;p
2	PLAN SCOPE MANAGEMENT INPUTS	has_input				[]		

Extract Relation

Entrée [47]:

```
my_grammar = r"""
NOUNS_VERB_NOUN: {<V.*>+<JJ.*>*<NN.*>+}"""
Sommaire['relation'] =""
for i in range(len(Sommaire)):
    if len(Sommaire['Contenu'][i]) != 0:
        text=Sommaire['Contenu'][i]
        tokens = nltk.word_tokenize(text)
        tags = nltk.pos_tag(tokens)
        text_parse_tree = get_parse_tree(my_grammar, tags)
        my_labels = get_labels_from_grammar(my_grammar)
        phrases = get_phrases_using_custom_labels(text_parse_tree, my_labels)
        for phrase in phrases:
            sentence =""
            for k in range(len(phrase)):
                sentence = sentence + phrase[k]+' '

            Sommaire['relation'][i] = Sommaire['relation'][i] + sentence + ';'

Sommaire
```

Out[47]:

	Process Name	Type	Contenu	Reference	Definition	Synonym	Sub
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	project s manage incl ;project
1	PLAN SCOPE MANAGEMENT		plan scope management process creating scope m...	depicted in Figure 5-2	Plan Scope Management is the process of creati...	[[plan, plans], [scope, process], [management,...	plan s ;manage pro cre ;pro
2	PLAN SCOPE MANAGEMENT INPUTS	has_input				[]	
3	PROJECT CHARTER		. the project charter documents project purpos...	Described in Section 4.1.3.1		[]	project ch docun pı purpose ;
4	PROJECT MANAGEMENT PLAN		. project management plan components include l...	Described in Section 8.1.3.1		[[management, managed]]	pı manage compor include
...	...	...	...	...	...	...	
258	WORK PERFORMANCE INFORMATION		. work performance information includes inform...	Described in Section 4.5.1.3		[]	perform inform incl ;infor

	Process Name	Type	Contenu	Reference	Definition	Synonym	Sub
259	COST FORECASTS		either calculated eac value bottomup eac value...			[]	docume communi
260	CHANGE REQUESTS		. analysis project performance may result chan...	Described in Section 4.3.3.4		[[requests, result], [requests, request]]	request sche base compor
261	PROJECT MANAGEMENT PLAN UPDATES		any change project management plan goes organi...	Described in Section 7.3.3.1	Any change to the project management plan goes...	[[project, projects]]	change pr manage plan ;organi
262	PROJECT DOCUMENTS UPDATES		project documents may updated result carrying ...	Described in Section 11.2.3.1	Project documents that may be updated as a res...	[[updates, updated]]	result car ;pro include li ;ne

263 rows × 9 columns

Entrée [48]:

```
my_grammar = r"""
NOUNS_VERB_NOUN: {<V.*>+}"""
Sommaire['Relation'] =""
for i in range(len(Sommaire)):
    if len(Sommaire['relation'][i]) != 0:
        text=Sommaire['relation'][i]
        tokens = nltk.word_tokenize(text)
        tags = nltk.pos_tag(tokens)
        text_parse_tree = get_parse_tree(my_grammar, tags)
        my_labels = get_labels_from_grammar(my_grammar)
        phrases = get_phrases_using_custom_labels(text_parse_tree, my_labels)
        for phrase in phrases:
            sentence =""
            for k in range(len(phrase)):
                sentence = sentence + phrase[k]+' '

            Sommaire['Relation'][i] = Sommaire['Relation'][i] + sentence + ';'
Sommaire
```

Out[48]:

	Process Name	Type	Contenu	Reference	Definition	Synonym	S
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	proje man ;pro
1	PLAN SCOPE MANAGEMENT		plan scope management process creating scope m...	depicted in Figure 5-2	Plan Scope Management is the process of creati...	[[plan, plans], [scope, process], [management,...	pl ;man
2	PLAN SCOPE MANAGEMENT INPUTS	has_input				[]	
3	PROJECT CHARTER		. the project charter documents project purpos...	Described in Section 4.1.3.1		[]	proje dc purpc
4	PROJECT MANAGEMENT PLAN		. project management plan components include l...	Described in Section 8.1.3.1		[[management, managed]]	man cor incl
...	...	...	...	...	...	...	
258	WORK PERFORMANCE INFORMATION		. work performance information includes inform...	Described in Section 4.5.1.3		[]	perf inf ;i

	Process Name	Type	Contenu	Reference	Definition	Synonym	S
259	COST FORECASTS		either calculated eac value bottomup eac value...			[]	doc comm
260	CHANGE REQUESTS		. analysis project performance may result chan...	Described in Section 4.3.3.4		[[requests, result], [requests, request]]	req con
261	PROJECT MANAGEMENT PLAN UPDATES		any change project management plan goes organi...	Described in Section 7.3.3.1	Any change to the project management plan goes...	[[project, projects]]	chang man ;or
262	PROJECT DOCUMENTS UPDATES		project documents may updated result carrying ...	Described in Section 11.2.3.1	Project documents that may be updated as a res...	[[updates, updated]]	result inclu

Extract object

Entrée [49]:

```
my_grammar = r"""
NOUNS_VERB_NOUN: {<V.*>+<JJ.*>*<NN.*>+}"""
Sommaire['Object1'] =""
for i in range(len(Sommaire)):
    if len(Sommaire['Contenu'][i]) != 0:
        text=Sommaire['Contenu'][i]
        tokens = nltk.word_tokenize(text)
        tags = nltk.pos_tag(tokens)
        text_parse_tree = get_parse_tree(my_grammar, tags)
        my_labels = get_labels_from_grammar(my_grammar)
        phrases = get_phrases_using_custom_labels(text_parse_tree, my_labels)
        for phrase in phrases:
            sentence =""
            for k in range(len(phrase)):
                sentence = sentence + phrase[k]+' '
            Sommaire['Object1'][i] = Sommaire['Object1'][i] + sentence + ';'
Sommaire
```

Out[49]:

	Process Name	Type	Contenu	Reference	Definition	Synonym	Sub
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	project s manage incl ;project
1	PLAN SCOPE MANAGEMENT		plan scope management process creating scope m...	depicted in Figure 5-2	Plan Scope Management is the process of creati...	[[plan, plans], [scope, process], [management,...	plan s ;manage pro cre ;pro
2	PLAN SCOPE MANAGEMENT INPUTS	has_input				[]	
3	PROJECT CHARTER		. the project charter documents project purpos...	Described in Section 4.1.3.1		[]	project ch docun pı purpose ;
4	PROJECT MANAGEMENT PLAN		. project management plan components include l...	Described in Section 8.1.3.1		[[management, managed]]	pı manage compoı include
...	...	...	...	...	...	...	
258	WORK PERFORMANCE INFORMATION		. work performance information includes inform...	Described in Section 4.5.1.3		[]	perform inform incl ;infor
259	COST FORECASTS		either calculated eac value bottomup eac value...			[]	docume communi

	Process Name	Type	Contenu	Reference	Definition	Synonym	Sub
260	CHANGE REQUESTS		. analysis project performance may result chan...	Described in Section 4.3.3.4		[[requests, result], [requests, request]]	request sche base compor
261	PROJECT MANAGEMENT PLAN UPDATES		any change project management plan goes organi...	Described in Section 7.3.3.1	Any change to the project management plan goes...	[[project, projects]]	change pr manage plan ;organi
262	PROJECT DOCUMENTS UPDATES		project documents may updated result carrying ...	Described in Section 11.2.3.1	Project documents that may be updated as a res...	[[updates, updated]]	result car ;pro include li ;ne

263 rows × 11 columns

Entrée [50]:

```
my_grammar = r"""
NOUNS_VERB_NOUN: {<NN.*>+}"""
Sommaire['object'] =""
for i in range(len(Sommaire)):
    if len(Sommaire['Object1'][i]) != 0:
        text=Sommaire['Object1'][i]
        tokens = nltk.word_tokenize(text)
        tags = nltk.pos_tag(tokens)
        text_parse_tree = get_parse_tree(my_grammar, tags)
        my_labels = get_labels_from_grammar(my_grammar)
        phrases = get_phrases_using_custom_labels(text_parse_tree, my_labels)
        for phrase in phrases:
            sentence =""
            for k in range(len(phrase)):
                sentence = sentence + phrase[k]+' '

            Sommaire['object'][i] = Sommaire['object'][i] + sentence + ';'
Sommaire
```

Out[50]:

	Process Name	Type	Contenu	Reference	Definition	Synonym	Subject1	su
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	project scope management includes ;project inc...	project manag ;project ;p
1	PLAN SCOPE MANAGEMENT		plan scope management process creating scope m...	depicted in Figure 5-2	Plan Scope Management is the process of creati...	[[plan, plans], [scope, process], [management,...	plan scope ;management process creating ;produ...	;manag pr ;p ;p
2	PLAN SCOPE MANAGEMENT INPUTS	has_input				[]		

Entrée [51]:

```
Sommaire.drop(['Subject1','relation','Object1'], axis=1, inplace=True)
```

Entrée [52]:

```
Sommaire["subject"] = Sommaire["subject"].str.split(";")
Sommaire["Relation"] = Sommaire["Relation"].str.split(";")
Sommaire["object"] = Sommaire["object"].str.split(";")
```

Entrée [108]:

```
Sommaire
```

Out[108]:

	Process Name	Type	Contenu	Reference	Definition	Synonym	sub
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	[project s manage , proj work
1	PLAN SCOPE MANAGEMENT		plan scope management process creating scope m...	depicted in Figure 5-2	Plan Scope Management is the process of creati...	[[plan, plans], [scope, process], [management,...	[p manage proc product s
2	PLAN SCOPE MANAGEMENT INPUTS	has_input				[]	
3	PROJECT CHARTER		. the project charter documents project purpos...	Described in Section 4.1.3.1		[]	[pr ch docume high proj
4	PROJECT MANAGEMENT PLAN		. project management plan components include l...	Described in Section 8.1.3.1		[[management, managed]]	[pr manage compone produ
...	...	...	...	...	...	...	
258	WORK PERFORMANCE INFORMATION		. work performance information includes inform...	Described in Section 4.5.1.3		[]	[perform informat inform
259	COST FORECASTS		either calculated eac value bottomup eac value...			[]	[valu
260	CHANGE REQUESTS		. analysis project performance may result chan...	Described in Section 4.3.3.4		[[requests, result], [requests, request]]	[request sche base compon
261	PROJECT MANAGEMENT PLAN UPDATES		any change project management plan goes organi...	Described in Section 7.3.3.1	Any change to the project management plan goes...	[[project, projects]]	[cha pr manage p organizati
262	PROJECT DOCUMENTS UPDATES		project documents may updated result carrying ...	Described in Section 11.2.3.1	Project documents that may be updated as a res...	[[updates, updated]]	[re proc assumpti producti

263 rows × 10 columns

Entrée [53]:

```
Sommaire['sentence']=''
for i in range(len(Sommaire)):
    for j in range(min([len(Sommaire['Relation'][i]) , len(Sommaire['subject'][i]), len(Som
        Sommaire['sentence'][i]= Sommaire['sentence'][i]+ '/'+ Sommaire['subject'][i][j] +
```

Entrée [110]:

```
Sommaire["sentence"]= Sommaire["sentence"].str.split("/")
Sommaire
```

Out[110]:

	Process Name	Type	Contenu	Reference	Definition	Synonym	sub
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	[project s manage , proj work
1	PLAN SCOPE MANAGEMENT		plan scope management process creating scope m...	depicted in Figure 5-2	Plan Scope Management is the process of creati...	[[plan, plans], [scope, process], [management,...	[p manage proc product s
2	PLAN SCOPE MANAGEMENT INPUTS	has_input				[]	
3	PROJECT CHARTER		. the project charter documents project purpos...	Described in Section 4.1.3.1		[]	[pr ch docume high proj
4	PROJECT MANAGEMENT PLAN		. project management plan components include l...	Described in Section 8.1.3.1		[[management, managed]]	[pr manage compone produ
...	...	...	...	...	...	...	
258	WORK PERFORMANCE INFORMATION		. work performance information includes inform...	Described in Section 4.5.1.3		[]	[perform informat inform
259	COST FORECASTS		either calculated eac value bottomup eac value...			[]	[valu
260	CHANGE REQUESTS		. analysis project performance may result chan...	Described in Section 4.3.3.4		[[requests, result], [requests, request]]	[request sche base compon
261	PROJECT MANAGEMENT PLAN UPDATES		any change project management plan goes organi...	Described in Section 7.3.3.1	Any change to the project management plan goes...	[[project, projects]]	[cha pr manage p organizati
262	PROJECT DOCUMENTS UPDATES		project documents may updated result carrying ...	Described in Section 11.2.3.1	Project documents that may be updated as a res...	[[updates, updated]]	[re proc assumpti producti

263 rows × 10 columns

Entrée [55]:

```
dfAux = Sommaire.explode('sentence')
dfAux
```

Out[55]:

	Process Name	Type	Contenu	Reference	Definition	Synonym	subject	Relation
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	[project scope management , project , work , p...	[includes , ensure , includes , managing , con...
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	[project scope management , project , work , p...	[includes , ensure , includes , managing , con...
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	[project scope management , project , work , p...	[includes , ensure , includes , managing , con...
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	[project scope management , project , work , p...	[includes , ensure , includes , managing , con...
0	PROJECT SCOPE MANAGEMENT		project scope management includes processes re...			[[project, projects], [scope, processes], [sco...	[project scope management , project , work , p...	[includes , ensure , includes , managing , con...

Entrée [56]:

```
dfAux.drop(['Process Name', 'Type', 'Contenu',
        'Reference', 'Definition',
        'Synonym',  'Relation' , 'subject' , 'object' ], axis=1, inplace=True)
```

Entrée [57]:

```
dfAux.dropna()
```

Out[57]:

	sentence
0	
0	project scope management ,includes ,processes
0	project ,ensure ,project
0	work ,includes ,work
0	plan ,managing ,project
...	...
262	budget variance analysis ,revisit ,basis estim...
262	value analysis ,reflect ,cost efficiency project
262	actions ,learned ,register
262	cost variances ,updated ,techniques

Entrée [58]:

```
dfAux=dfAux.join(dfAux['sentence'].str.split(',',  expand=True).rename(columns={0:'Subject'
```

Entrée [59]:

```
dfAux=dfAux.mask(dfAux.eq('None')).dropna()
```

Entrée [60]:

```
dfAux=dfAux.drop_duplicates()
```

Entrée [61]:

```
dfAux.drop('sentence',axis=1)
```

Out[61]:

	Subject	Relation	Object
0	project scope management	includes	processes
0	project	ensure	project
0	work	includes	work
0	plan	managing	project
0	managementthe	concerned defining controlling	project scope
...	...	...	...
262	budget variance analysis	revisit	basis estimates
262	value analysis	reflect	cost efficiency project
262	actions	learned	register
262	cost variances	updated	techniques

Entrée [62]:

```
for i in range(len(Sommaire)):
    Sommaire['Contenu'][i] = Sommaire['Contenu'][i].replace('part gui', '')
```

Entrée [63]:

```
sousT = d1.copy()
ST = d2.copy()
SST = d3.copy()

SST = SST.drop(['start'], axis=1)
SST.insert(0,column='Process_Name',value="")
SST.insert(2,column='Type',value="")
SST.insert(3,column='Definition',value="")
```

Entrée [64]:

```
for i in range(len(SST)):
    SST['Process_Name'][i] = SST['Process Name'][i][0:5]

for i in range(len(SST)):
    for j in range(len(ST)):
        if ST['Process Name'][j].startswith(SST['Process_Name'][i]):
            SST['Process_Name'][i] = ST['Process Name'][j]
SST
```

Out[64]:

	Process_Name	Process Name	Type	Definition
0	5.1.1 PLAN SCOPE MANAGEMENT: INPUTS	5.1.1.1 PROJECT CHARTER		
1	5.1.1 PLAN SCOPE MANAGEMENT: INPUTS	5.1.1.2 PROJECT MANAGEMENT PLAN		
2	5.1.1 PLAN SCOPE MANAGEMENT: INPUTS	5.1.1.3 ENTERPRISE ENVIRONMENTAL FACTORS		
3	5.1.1 PLAN SCOPE MANAGEMENT: INPUTS	5.1.1.4 ORGANIZATIONAL PROCESS ASSETS		
4	5.1.2 PLAN SCOPE MANAGEMENT: TOOLS AND TECHNIQ...	5.1.2.1 EXPERT JUDGMENT		
...	...	...	...	...
191	7.4.3 CONTROL COSTS: OUTPUTS	7.4.3.1 WORK PERFORMANCE INFORMATION		
192	7.4.3 CONTROL COSTS: OUTPUTS	7.4.3.2 COST FORECASTS		
193	7.4.3 CONTROL COSTS: OUTPUTS	7.4.3.3 CHANGE REQUESTS		
194	7.4.3 CONTROL COSTS: OUTPUTS	7.4.3.4 PROJECT MANAGEMENT PLAN UPDATES		
195	7.4.3 CONTROL COSTS: OUTPUTS	7.4.3.5 PROJECT DOCUMENTS UPDATES		

196 rows × 4 columns

Entrée [65]:

```
for i in range(len(SST)):
    if 'INPUTS' in SST['Process_Name'][i]:
        SST['Type'][i] = 'HAS_INPUTS'
    elif 'OUTPUTS' in SST['Process_Name'][i]:
        SST['Type'][i] = 'HAS_OUTPUTS'
    else:
        SST['Type'][i] = 'HAS_TOOLS_AND_TECHNIQUES'
```

Entrée [66]:

```
for i in range(len(SST)):
    SST['Process_Name'][i] = SST['Process_Name'][i][ 6 : SST['Process_Name'][i].index(":")
    SST['Process Name'][i] = SST['Process Name'][i][8:]
SST
```

Out[66]:

	Process_Name	Process Name	Type	Definition
0	PLAN SCOPE MANAGEMENT	PROJECT CHARTER	HAS_INPUTS	
1	PLAN SCOPE MANAGEMENT	PROJECT MANAGEMENT PLAN	HAS_INPUTS	
2	PLAN SCOPE MANAGEMENT	ENTERPRISE ENVIRONMENTAL FACTORS	HAS_INPUTS	
3	PLAN SCOPE MANAGEMENT	ORGANIZATIONAL PROCESS ASSETS	HAS_INPUTS	
4	PLAN SCOPE MANAGEMENT	EXPERT JUDGMENT	HAS_TOOLS_AND_TECHNIQUES	
...	...	...	...	...
191	CONTROL COSTS	WORK PERFORMANCE INFORMATION	HAS_OUTPUTS	
192	CONTROL COSTS	COST FORECASTS	HAS_OUTPUTS	
193	CONTROL COSTS	CHANGE REQUESTS	HAS_OUTPUTS	
194	CONTROL COSTS	PROJECT MANAGEMENT PLAN UPDATES	HAS_OUTPUTS	
195	CONTROL COSTS	PROJECT DOCUMENTS UPDATES	HAS_OUTPUTS	

196 rows × 4 columns

Entrée [67]:

```
for i in range(len(SST)):
    for j in range(len(Sommaire)):
        if SST['Process Name'][i].startswith(Sommaire['Process Name'][j]):
            if len(Sommaire['Definition'][j]) == 0 :
                SST['Definition'][i] = Sommaire['Contenu'][j]
            else :
                SST['Definition'][i] = Sommaire['Definition'][j]
```

Entrée [68]:

```
SST = SST.rename({'Process Name': 'Concept'}, axis=1)
SST
```

Out[68]:

	Process_Name	Concept	Type	Definition
0	PLAN SCOPE MANAGEMENT	PROJECT CHARTER	HAS_INPUTS	. the project charter provides preapproved fin...
1	PLAN SCOPE MANAGEMENT	PROJECT MANAGEMENT PLAN	HAS_INPUTS	. project management plan components include l...
2	PLAN SCOPE MANAGEMENT	ENTERPRISE ENVIRONMENTAL FACTORS	HAS_INPUTS	the enterprise environmental factors influence...
3	PLAN SCOPE MANAGEMENT	ORGANIZATIONAL PROCESS ASSETS	HAS_INPUTS	the organizational process assets influence co...
4	PLAN SCOPE MANAGEMENT	EXPERT JUDGMENT	HAS_TOOLS_AND_TECHNIQUES	. examples expert judgment control costs proce...
...	...	...	...	...
191	CONTROL COSTS	WORK PERFORMANCE INFORMATION	HAS_OUTPUTS	. work performance information includes inform...
192	CONTROL COSTS	COST FORECASTS	HAS_OUTPUTS	either calculated eac value bottomup eac value...
193	CONTROL COSTS	CHANGE REQUESTS	HAS_OUTPUTS	. analysis project performance may result chan...
194	CONTROL COSTS	PROJECT MANAGEMENT PLAN UPDATES	HAS_OUTPUTS	Any change to the project management plan goes...
195	CONTROL COSTS	PROJECT DOCUMENTS UPDATES	HAS_OUTPUTS	Project documents that may be updated as a res...

196 rows × 4 columns

Entrée [69]:

```
SST.to_excel("converted-to-excel.xlsx")
```

OWL Generation

Entrée [70]:

```
from rdflib.namespace import DC, DCTERMS, DOAP, FOAF, OWL, RDF, RDFS, SKOS, VOID, XMLNS, XS
from rdflib import URIRef, BNode, Literal, Namespace, Graph
from rdflib.extras import describer
```

Entrée [71]:

```
g= Graph()
g.bind("owl",OWL)
g.bind("pr","http://example.org/projectOntology/")
ns_url = "http://example.org/projectOntology/"
g.add((URIRef('http://example.org/projectOntology/'), RDF.type, OWL.Ontology ))
```

Out[71]:

```
<Graph identifier=N2b83637dcc4c45a18cf4278a48de8ba0 (<class 'rdflib.graph.Gr
aph'>)>
```

Entrée [72]:

```
df_process=SST.drop_duplicates(subset='Process_Name', keep='first', inplace=False)
df_process=df_process['Process_Name']
df_process
```

Out[72]:

```
0             PLAN SCOPE MANAGEMENT
9              COLLECT REQUIREMENTS
26                     DEFINE SCOPE
38                       CREATE WBS
46                    VALIDATE SCOPE
56                     CONTROL SCOPE
65          PLAN SCHEDULE MANAGEMENT
73                 DEFINE ACTIVITIES
85              SEQUENCE ACTIVITIES
95      ESTIMATE ACTIVITY DURATIONS
110                DEVELOP SCHEDULE
130                CONTROL SCHEDULE
145            PLAN COST MANAGEMENT
153                  ESTIMATE COSTS
168                DETERMINE BUDGET
183                   CONTROL COSTS
Name: Process_Name, dtype: object
```

Entrée [73]:

```
for c in df_process:

    cl = URIRef(ns_url+c.replace(" ","_"))
    g.add((cl, RDF.type, OWL.Class))
    clOutput =URIRef(ns_url+c.replace(" ","_")+"_"+"Outputs")
    g.add((clOutput, RDF.type, OWL.Class))
    clTools =URIRef(ns_url+c.replace(" ","_")+"_"+"Tools and techniques".replace(" ","_"))
    g.add((clTools, RDF.type,OWL.Class))
    clInput =URIRef(ns_url+c.replace(" ","_")+"_"+"Inputs".replace(" ","_"))
    g.add((clInput, RDF.type, OWL.Class))

for o in SST.index :
    for c in df_process:
        if c == SST.loc[o,'Process_Name'] and  SST.loc[o,'Type'] == 'HAS_OUTPUTS' :

            clo = URIRef(ns_url+c.replace(" ","_")+"_"+"Outputs".replace(" ","_"))
            ind = URIRef(ns_url+SST.loc[o,'Concept'].replace(" ","_"))
            g.add((ind, RDF.type, clo))


        if  c == SST.loc[o,'Process_Name'] and SST.loc[o,'Type'] == 'HAS_INPUTS' :


            clo = URIRef(ns_url+c.replace(" ","_")+"_"+"Inputs".replace(" ","_"))
            ind = URIRef(ns_url+SST.loc[o,'Concept'].replace(" ","_"))
            g.add((ind, RDF.type, clo))

        if c == SST.loc[o,'Process_Name'] and SST.loc[o,'Type'] == 'HAS_TOOLS_AND_TECHNIQUE


            clo = URIRef(ns_url+c.replace(" ","_")+"_"+"Tools and techniques".replace(" ","
            ind = URIRef(ns_url+SST.loc[o,'Concept'].replace(" ","_"))
            g.add((ind, RDF.type, clo))
```

Adding Object Property

Entrée [74]:

```
for i in SST.index :
    if SST.loc[i,'Type'] == 'HAS_TOOLS_AND_TECHNIQUES' :

        c = URIRef(ns_url+SST.loc[i,'Type'].replace(" ","_"))
        domaine = URIRef(ns_url+SST.loc[i,'Process_Name'].replace(" ","_"))
        rang = URIRef(ns_url+SST.loc[i,'Process_Name'].replace(" ","_")+"_Tools and techniq
    if SST.loc[i,'Type'] == 'HAS_INPUTS' :

        c = URIRef(ns_url+SST.loc[i,'Type'].replace(" ","_"))
        domaine = URIRef(ns_url+SST.loc[i,'Process_Name'].replace(" ","_"))
        rang = URIRef(ns_url+SST.loc[i,'Process_Name'].replace(" ","_")+"_Inputs".replace("

    if SST.loc[i,'Type'] == 'HAS_OUTPUTS' :

        c = URIRef(ns_url+SST.loc[i,'Type'].replace(" ","_"))
        domaine = URIRef(ns_url+SST.loc[i,'Process_Name'].replace(" ","_"))
        rang = URIRef(ns_url+SST.loc[i,'Process_Name'].replace(" ","_")+"_Outputs".replace(


    g.add((c, RDF.type, OWL.ObjectProperty))
    g.add((c, RDFS.domain, domaine))
    g.add((c, RDFS.range, rang))
```

Adding Annotation isDefinedBy

Entrée [75]:

```
process_definition = pd.DataFrame(columns=['Process_Name','Definition'])
for i in range(len(SST)):
    process_definition = process_definition.append({'Process_Name' : SST['Process_Name'][i]
```

Entrée [76]:

```
process_definition = process_definition.drop_duplicates()
process_definition
```

Out[76]:

	Process_Name	Definition
0	PLAN SCOPE MANAGEMENT	NaN
9	COLLECT REQUIREMENTS	NaN
26	DEFINE SCOPE	NaN
38	CREATE WBS	NaN
46	VALIDATE SCOPE	NaN
56	CONTROL SCOPE	NaN
65	PLAN SCHEDULE MANAGEMENT	NaN
73	DEFINE ACTIVITIES	NaN
85	SEQUENCE ACTIVITIES	NaN
95	ESTIMATE ACTIVITY DURATIONS	NaN
110	DEVELOP SCHEDULE	NaN
130	CONTROL SCHEDULE	NaN
145	PLAN COST MANAGEMENT	NaN
153	ESTIMATE COSTS	NaN
168	DETERMINE BUDGET	NaN
183	CONTROL COSTS	NaN

Entrée [77]:

```
process_definition = process_definition. reset_index ( drop = True )
```

Entrée [78]:

```
for i in range(len(Sommaire)):
    for j in range(len(process_definition)):
        if process_definition['Process_Name'][j] == Sommaire['Process Name'][i]:
            process_definition['Definition'][j] = Sommaire['Definition'][i]
```

Entrée [79]:

```
process_definition
```

Out[79]:

	Process_Name	Definition
0	PLAN SCOPE MANAGEMENT	Plan Scope Management is the process of creati...
1	COLLECT REQUIREMENTS	Collect Requirements is the process of determi...
2	DEFINE SCOPE	Define Scope is the process of developing a de...
3	CREATE WBS	Create WBS is the process of subdividing proje...
4	VALIDATE SCOPE	Validate Scope is the process of formalizing a...
5	CONTROL SCOPE	Control Scope is the process of monitoring the...
6	PLAN SCHEDULE MANAGEMENT	Plan Schedule Management is the process of est...
7	DEFINE ACTIVITIES	Define Activities is the process of identifyin...
8	SEQUENCE ACTIVITIES	Sequence Activities is the process of identify...
9	ESTIMATE ACTIVITY DURATIONS	Estimate Activity Durations is the process of ...
10	DEVELOP SCHEDULE	Develop Schedule is the process of analyzing a...
11	CONTROL SCHEDULE	Control Schedule is the process of monitoring ...
12	PLAN COST MANAGEMENT	Plan Cost Management is the process of definin...
13	ESTIMATE COSTS	Estimate Costs is the process of developing an...
14	DETERMINE BUDGET	Determine Budget is the process of aggregating...
15	CONTROL COSTS	Control Costs is the process of monitoring the...

Entrée [80]:

```
for i in range(len(process_definition)) :
    c = URIRef(ns_url+process_definition.loc[i,'Process_Name'].replace(" ","_"))
    desc=process_definition.loc[i,'Definition']
    definedby = Literal(desc,datatype=XSD.string)
    g.add((c, RDFS.isDefinedBy, definedby))
```

Entrée [81]:

```
SST = SST.reset_index(drop=True)
```

Entrée [82]:

```
for i in range(len(SST)) :
    c = URIRef(ns_url+SST.loc[i,'Concept'].replace(" ","_"))
    desc=SST.loc[i,'Definition']
    definedby = Literal(desc,datatype=XSD.string)
    g.add((c, RDFS.isDefinedBy, definedby))
```

Adding Annotation seeAlso

Entrée [83]:

```
# df_process_reset=df_process.reset_index(drop=True)
```

Entrée [84]:

```
# for i in range(len(df_process_reset)):
#     c = URIRef(ns_url+df_process_reset[i].replace(" ","_"))
#     for j in range(len(bigD[df_process_reset[i]])):
#         syn = Literal( bigD[df_process_reset[i]][j] ,datatype=XSD.string)
#         g.add((c, RDFS.seeAlso, syn))
```

Entrée [85]:

```
# from collections import defaultdict
# # bigD = defaultdict(list)
```

Entrée [86]:

```
# final_concepts = []
# # final_concepts.extend(SST['Concept'].drop_duplicates().tolist())
```

Entrée [87]:

```
# for j in range(len(Sommaire)):
#     for i in range(len(final_concepts)):
#         if Sommaire['Process Name'][j] in final_concepts[i]:
#             bigD[final_concepts[i]] = Sommaire['Synonym'][j]
```

Adding DataProperty

concept ref w nzid col range te5ou mel ref

Entrée [88]:

```
final_concepts = []
final_concepts.extend(SST['Concept'].drop_duplicates().tolist())
```

Entrée [89]:

```
df_dataproperty2 = pd.DataFrame(columns=['Domain','Data_Property' , 'Range'])
```

Entrée [90]:

```
for i in range(len(Sommaire)):
    for j in range(len(final_concepts)):
        if Sommaire['Process Name'][i] in final_concepts[j]:
            if len(Sommaire['Reference'][i]) != 0:
                df_dataproperty2 =  df_dataproperty2.append({'Domain' : final_concepts[j] ,
```

Entrée [91]:

```
df_dataproperty2
```

Out[91]:

	Domain	Data_Property	Range
0	PROJECT CHARTER	Described in	Section 4.1.3.1
1	PROJECT MANAGEMENT PLAN	Described in	Section 8.1.3.1
2	PROJECT MANAGEMENT PLAN UPDATES	Described in	Section 8.1.3.1
3	EXPERT JUDGMENT	Described in	Section 4.1.2.1
4	PROJECT CHARTER	Described in	Section 4.1.3.1
...	...	...	...
121	DATA ANALYSIS	Described in	Section 7.2.2.6
122	WORK PERFORMANCE INFORMATION	Described in	Section 4.5.1.3
123	CHANGE REQUESTS	Described in	Section 4.3.3.4
124	PROJECT MANAGEMENT PLAN UPDATES	Described in	Section 7.3.3.1
125	PROJECT DOCUMENTS UPDATES	Described in	Section 11.2.3.1

126 rows × 3 columns

Entrée [92]:

```
df_testsr = df_dataproperty2['Domain'].str.lower()
```

Entrée [93]:

```
df_testsr = df_dataproperty2['Range'].str.lower()
```

Entrée [94]:

```
for i in range(len(df_dataproperty2)):
    c = URIRef(ns_url+df_dataproperty2.loc[i,'Data_Property'].replace(" ","_"))
    domaine = URIRef(ns_url+df_dataproperty2.loc[i,'Domain'].replace(" ","_"))
    g.add((c, RDF.type, OWL.DatatypeProperty))
    g.add((c, RDFS.domain, domaine))
    g.add((c, RDFS.range, XSD.string))
```

Entrée [95]:

```
for i in range(len(df_dataproperty2)):
    indiv = URIRef(ns_url+df_dataproperty2.loc[i,'Domain'].replace(" ","_"))
    section = Literal(df_dataproperty2.loc[i,'Range'],datatype=XSD.string)
    data_prop=URIRef(ns_url+df_dataproperty2.loc[i,'Data_Property'].replace(" ","_"))

    g.add((indiv,data_prop,section))
```

Entrée [96]:

```
g.serialize(destination='ontologyfinal.owl', format='turtle')
```

Out[96]:

```
<Graph identifier=N2b83637dcc4c45a18cf4278a48de8ba0 (<class 'rdflib.graph.Gr
aph'>)>
```

Entrée [97]:

```
with open('ontologyfinal.owl', 'w', encoding='utf-8') as output:
    output.write(g.serialize(format="pretty-xml"))
```

Extract information from owl

Entrée [98]:

```
from owlready2 import *
import pandas as pd
onto = get_ontology("ontologyfinal.owl").load()
```

```
* Owlready2 * Warning: optimized Cython parser module 'owlready2_optimized'
is not available, defaulting to slower Python implementation
```

Entrée [99]:

```
#Get classes
list(onto.object_properties())
```

Out[99]:

```
[ontologyfinal.HAS_OUTPUTS,
 ontologyfinal.HAS_INPUTS,
 ontologyfinal.HAS_TOOLS_AND_TECHNIQUES]
```

Entrée [100]:

```
list(onto.data_properties())
```

Out[100]:

```
[ontologyfinal.Described_in,
 ontologyfinal.described_in,
 ontologyfinal.shown_in]
```

Entrée [101]:

```
list(onto.properties())
```

Out[101]:

```
[ontologyfinal.Described_in,
 ontologyfinal.described_in,
 ontologyfinal.shown_in,
 ontologyfinal.HAS_OUTPUTS,
 ontologyfinal.HAS_INPUTS,
 ontologyfinal.HAS_TOOLS_AND_TECHNIQUES]
```

Entrée [102]:

```
list(onto.classes())
```

Out[102]:

```
[ontologyfinal.PLAN_SCOPE_MANAGEMENT,
 ontologyfinal.COLLECT_REQUIREMENTS,
 ontologyfinal.DEFINE_SCOPE,
 ontologyfinal.CREATE_WBS,
 ontologyfinal.VALIDATE_SCOPE,
 ontologyfinal.CONTROL_SCOPE,
 ontologyfinal.PLAN_SCHEDULE_MANAGEMENT,
 ontologyfinal.DEFINE_ACTIVITIES,
 ontologyfinal.SEQUENCE_ACTIVITIES,
 ontologyfinal.ESTIMATE_ACTIVITY_DURATIONS,
 ontologyfinal.DEVELOP_SCHEDULE,
 ontologyfinal.CONTROL_SCHEDULE,
 ontologyfinal.PLAN_COST_MANAGEMENT,
 ontologyfinal.ESTIMATE_COSTS,
 ontologyfinal.DETERMINE_BUDGET,
 ontologyfinal.CONTROL_COSTS,
 ontologyfinal.PLAN_SCOPE_MANAGEMENT_Outputs,
 ontologyfinal.COLLECT_REQUIREMENTS_Outputs,
 ontologyfinal.DEFINE_SCOPE_Outputs,
 ontologyfinal.CREATE_WBS_Outputs,
 ontologyfinal.VALIDATE_SCOPE_Outputs,
 ontologyfinal.CONTROL_SCOPE_Outputs,
 ontologyfinal.PLAN_SCHEDULE_MANAGEMENT_Outputs,
 ontologyfinal.DEFINE_ACTIVITIES_Outputs,
 ontologyfinal.SEQUENCE_ACTIVITIES_Outputs,
 ontologyfinal.ESTIMATE_ACTIVITY_DURATIONS_Outputs,
 ontologyfinal.DEVELOP_SCHEDULE_Outputs,
 ontologyfinal.CONTROL_SCHEDULE_Outputs,
 ontologyfinal.PLAN_COST_MANAGEMENT_Outputs,
 ontologyfinal.ESTIMATE_COSTS_Outputs,
 ontologyfinal.DETERMINE_BUDGET_Outputs,
 ontologyfinal.CONTROL_COSTS_Outputs,
 ontologyfinal.PLAN_SCOPE_MANAGEMENT_Inputs,
 ontologyfinal.COLLECT_REQUIREMENTS_Inputs,
 ontologyfinal.DEFINE_SCOPE_Inputs,
 ontologyfinal.CREATE_WBS_Inputs,
 ontologyfinal.VALIDATE_SCOPE_Inputs,
 ontologyfinal.CONTROL_SCOPE_Inputs,
 ontologyfinal.PLAN_SCHEDULE_MANAGEMENT_Inputs,
 ontologyfinal.DEFINE_ACTIVITIES_Inputs,
 ontologyfinal.SEQUENCE_ACTIVITIES_Inputs,
 ontologyfinal.ESTIMATE_ACTIVITY_DURATIONS_Inputs,
 ontologyfinal.DEVELOP_SCHEDULE_Inputs,
 ontologyfinal.CONTROL_SCHEDULE_Inputs,
 ontologyfinal.PLAN_COST_MANAGEMENT_Inputs,
 ontologyfinal.ESTIMATE_COSTS_Inputs,
 ontologyfinal.DETERMINE_BUDGET_Inputs,
 ontologyfinal.CONTROL_COSTS_Inputs,
 ontologyfinal.PLAN_SCOPE_MANAGEMENT_Tools_and_techniques,
 ontologyfinal.COLLECT_REQUIREMENTS_Tools_and_techniques,
 ontologyfinal.DEFINE_SCOPE_Tools_and_techniques,
 ontologyfinal.CREATE_WBS_Tools_and_techniques,
 ontologyfinal.PLAN_SCHEDULE_MANAGEMENT_Tools_and_techniques,
 ontologyfinal.DEFINE_ACTIVITIES_Tools_and_techniques,
 ontologyfinal.ESTIMATE_ACTIVITY_DURATIONS_Tools_and_techniques,
```

```
 ontologyfinal.PLAN_COST_MANAGEMENT_Tools_and_techniques,
 ontologyfinal.ESTIMATE_COSTS_Tools_and_techniques,
 ontologyfinal.DETERMINE_BUDGET_Tools_and_techniques,
 ontologyfinal.CONTROL_COSTS_Tools_and_techniques,
 ontologyfinal.CONTROL_SCOPE_Tools_and_techniques,
 ontologyfinal.DEVELOP_SCHEDULE_Tools_and_techniques,
 ontologyfinal.CONTROL_SCHEDULE_Tools_and_techniques,
 ontologyfinal.VALIDATE_SCOPE_Tools_and_techniques,
 ontologyfinal.SEQUENCE_ACTIVITIES_Tools_and_techniques]
```

Entrée [103]:

```
#Get instances
for classe in onto.classes():
    for subc in classe.instances():
        print(classe ,"=>" , subc)
```

```
ontologyfinal.PLAN_SCOPE_MANAGEMENT_Outputs => ontologyfinal.SCOPE_MANAGEM
ENT_PLAN_
ontologyfinal.PLAN_SCOPE_MANAGEMENT_Outputs => ontologyfinal.REQUIREMENTS_
MANAGEMENT_PLAN_
ontologyfinal.COLLECT_REQUIREMENTS_Outputs => ontologyfinal.REQUIREMENTS_T
RACEABILITY_MATRIX_
ontologyfinal.COLLECT_REQUIREMENTS_Outputs => ontologyfinal.REQUIREMENTS_D
OCUMENTATION_
ontologyfinal.DEFINE_SCOPE_Outputs => ontologyfinal.PROJECT_DOCUMENTS_UPDA
TES_
ontologyfinal.DEFINE_SCOPE_Outputs => ontologyfinal.PROJECT_SCOPE_STATEMEN
T_
ontologyfinal.CREATE_WBS_Outputs => ontologyfinal.PROJECT_DOCUMENTS_UPDATE
S_
ontologyfinal.CREATE_WBS_Outputs => ontologyfinal.SCOPE_BASELINE_
ontologyfinal.VALIDATE_SCOPE_Outputs => ontologyfinal.PROJECT_DOCUMENTS_UP
DATES_
ontologyfinal.VALIDATE_SCOPE_Outputs => ontologyfinal.CHANGE_REQUESTS_
ontologyfinal.VALIDATE_SCOPE_Outputs => ontologyfinal.WORK_PERFORMANCE_INF
```

Entrée [104]:

```
requete = "PLAN_SCOPE_MANAGEMENT"
#Get Properties From OWL
# t_inputs = pd.DataFrame(columns=['inputs', 'Subclasse' , 'property'])
# c = ""
# s = ""
# p = ""
t_outputs = pd.DataFrame(columns=['outputs', 'Subclasse' , 'property'])
for classe in list(onto.classes()):
    if requete.upper() in str(classe).upper():
        if ('_Outputs') in str(classe):
            for subc in classe.instances():
                for prop in subc.get_properties():
                    for value in prop[subc]:
                        t_outputs = t_outputs.append({ 'outputs':str(classe).replace("ontol

#                         ch =   "classe:" + str(classe) + "\n" +"subclasse :" + str(subc)+
#                         tab.append(ch)
t_outputs
```

Out[104]:

	outputs	Subclasse	property
0	PLAN_SCOPE_MANAGEMENT_Outputs	SCOPE_MANAGEMENT_PLAN_	isDefinedBy the scope management plan componen...
1	PLAN_SCOPE_MANAGEMENT_Outputs	REQUIREMENTS_MANAGEMENT_PLAN_	isDefinedBy the requirements management plan c...

Entrée [105]:

```
requete = "PLAN_SCOPE_MANAGEMENT"
t_inputs = pd.DataFrame(columns=['inputs', 'Subclasse', 'property'])
for classe in list(onto.classes()):
    if requete.upper() in str(classe).upper():
        if ('_Inputs') in str(classe):
            for subc in classe.instances():
                for prop in subc.get_properties():
                    for value in prop[subc]:
                        t_inputs = t_inputs.append({'inputs': str(classe).replace("ontology
                                                    'Subclasse': str(subc).replace("ont
                                                    'property': str(prop.python_name) +
                                                   ignore_index=True)
t_inputs
```

Out[105]:

	inputs	Subclasse	properl
0	PLAN_SCOPE_MANAGEMENT_Inputs	PROJECT_CHARTER_	isDefinedB the proje charte provide prea.
1	PLAN_SCOPE_MANAGEMENT_Inputs	PROJECT_CHARTER_	Described_ Section 4.1.3.
2	PLAN_SCOPE_MANAGEMENT_Inputs	PROJECT_CHARTER_	Described_ Section 4.2.3.
3	PLAN_SCOPE_MANAGEMENT_Inputs	PROJECT_MANAGEMENT_PLAN_	isDefinedB proje manageme pla component.
4	PLAN_SCOPE_MANAGEMENT_Inputs	PROJECT_MANAGEMENT_PLAN_	Described_ Section 8.1.3.
5	PLAN_SCOPE_MANAGEMENT_Inputs	PROJECT_MANAGEMENT_PLAN_	Described_ Section 5.1.3.
6	PLAN_SCOPE_MANAGEMENT_Inputs	PROJECT_MANAGEMENT_PLAN_	Described_ Section 4.2.3.
7	PLAN_SCOPE_MANAGEMENT_Inputs	PROJECT_MANAGEMENT_PLAN_	Described_ Section 5.4.3.
8	PLAN_SCOPE_MANAGEMENT_Inputs	PROJECT_MANAGEMENT_PLAN_	Described_ Sectic 11.1.3.
9	PLAN_SCOPE_MANAGEMENT_Inputs	PROJECT_MANAGEMENT_PLAN_	Described_ Section 4.4.3.
10	PLAN_SCOPE_MANAGEMENT_Inputs	ENTERPRISE_ENVIRONMENTAL_FACTORS_	isDefinedByth enterpris environment factor.
11	PLAN_SCOPE_MANAGEMENT_Inputs	ORGANIZATIONAL_PROCESS_ASSETS_	isDefinedByth organization process asse i.

Entrée [106]:

```
requete = "PLAN_SCOPE_MANAGEMENT"
su = "PROJECT_CHARTER_"
t_inputs = pd.DataFrame(columns=['inputs', 'Subclasse', 'property'])
for classe in list(onto.classes()):
    if requete.upper() in str(classe).upper():
        if ('_Inputs') in str(classe):
            for subc in classe.instances():
                if su in str(subc):
                    for prop in subc.get_properties():
                        for value in prop[subc]:
                            t_inputs = t_inputs.append({'inputs': str(classe).replace("onto
                                                        'Subclasse': str(subc).replace("ont
                                                        'property': str(prop.python_name) +
                                                       ignore_index=True)
# t_inputs
d = {}
tabProp = []
for i in range(len(t_inputs)):
    tabProp.append(t_inputs['property'][i])
    d[t_inputs['Subclasse'][i]]=tabProp
d
```

Out[106]:

```
{'PROJECT_CHARTER_': ['isDefinedBy. the project charter provides preapproved
financial resources detailed project costs developed . the project charter a
lso defines project approval requirements influence management project costs
',
  'Described_in Section 4.1.3.1',
  'Described_in Section 4.2.3.1']}
```

Entrée [107]:

```
requete = "PLAN_SCOPE_MANAGEMENT"
#Get Properties From OWL
t_ToolsAndTech = pd.DataFrame(columns=['ToolsAndTech', 'Subclasse' , 'property'])
for classe in list(onto.classes()):
    if requete.upper() in str(classe).upper():
        if ('_Tools_and_techniques') in str(classe):
            for subc in classe.instances():
                for prop in subc.get_properties():
                    for value in prop[subc]:
                        t_ToolsAndTech = t_ToolsAndTech.append({ 'ToolsAndTech':str(classe)
d2 = {}
tabProp2 = []
for i in range(len(t_ToolsAndTech)):
    try:
        if t_ToolsAndTech['Subclasse'][i] == t_ToolsAndTech['Subclasse'][i-1]:
            tabProp.append(t_ToolsAndTech['property'][i])
        else:
            d2[t_ToolsAndTech['Subclasse'][i-1]]=tabProp2
            tabProp2 = []
            tabProp2.append(t_ToolsAndTech['property'][i])
    except:
        tabProp2.append(t_ToolsAndTech['property'][i])
d2
```

Out[107]:

```
{'EXPERT_JUDGMENT_': ['isDefinedBy. examples expert judgment control costs
process include limited variance analysis earned v alue analysis forecasti
ng financial analysis '],
 'DATA_ANALYSIS_': ['isDefinedByData analysis techniques that can be used
to control costs include but are not limited to Earned value analysis EVA.
Earned value analysis compares the performance measurement baseline to the
actual schedule and cost performance. EVM integrates the scope baseline wi
th the cost baseline and schedule baseline to form the performance measure
ment baseline. EVM develops and monitors three key dimensions for each wor
k package and control account  Planned value. Planned value PV is the auth
orized budget assigned to scheduled work. It is the authorized budget plan
ned for the work to be accomplished for an activity or work breakdown stru
cture WBS component not including management reserve. This budget is  Part
Guide allocated by phase over the life of the project but at a given point
in time planned value defines the physical work that should have been acco
mplished. The total of the PV is sometimes referred to as the performance
measurement baseline PMB. The total planned value for the project is also
known as budget at completion BAC.  Earned value. Earned value EV is a mea
sure of work performed expressed in terms of the budget authorized for tha
t work. It is the budget associated with the authorized work that has been
completed. The EV being measured needs to be related to the PMB and the EV
measured cannot be greater than the authorized PV budget for a component.
The EV is often used to calculate the percent complete of a project. Progr
ess measurement criteria should be established for each WBS component to m
easure work in progress. Project managers monitor EV both incrementally to
determine current status and cumulatively to determine the longterm perfor
mance trends.  Actual cost. Actual cost AC is the realized cost incurred f
or the work performed on an activity during a specific time period. It is
the total cost incurred in accomplishing the work that the EV measured. Th
e AC needs to correspond in definition to what was budgeted in the PV and
measured in the EV e.g. direct hours only direct costs only or all costs i
```

```
ncluding indirect costs. The AC will have no upper limit whatever is spent
to achieve the EV will be measured. Variance analysis. ']}
```