

Rapport

Projet Data Science

SPÉCIALITÉ : INFORMATIQUE

Elaboré par :

Sarra Ferchichi

MedYassine Ben Nacef

Mayssa Zaouali

Yessine Khanfir

Fedi lahbib

Wassim yaich

Khalil jeber

5DS2

Supervisé par : Mme Wiem Zaouga

Table des matières

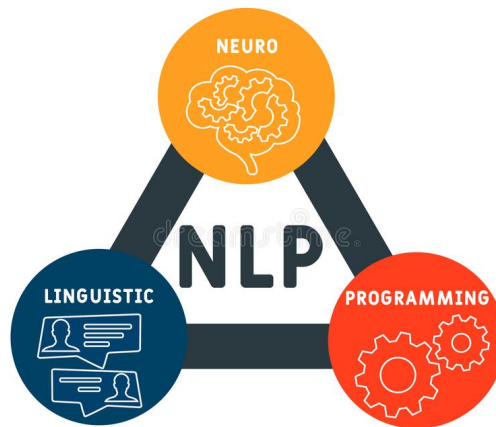
Table des matières	2
1.Définition NLP :	3
2.Source des données :	4
3.Les phases du prétraitement NLP :	4
3.1.Nettoyage :	4
3.2.Normalisation et Exploration des données:	5
4.TF-IDF	8
5. Named Entities Extraction:	10
6. Cosine Similarity Measure:	11
7. Latent Dirichlet Allocation:	11
8. Extraction des relations:	13
8.1.Extraction des classes :	17
8.2.Extraction des subclasses :	17
9. Ontology learning process:	18
9.1 Processus de construction de composants OWL	19
9.1.1.RDF(Resource Description Framework) :	19
9.1.2. : Ajout des classes	19
9.1.3. : Ajout des subclasses	20
9.1.4. : Ajout des individuals	20
9.1.5. : Ajout des object properties	20
9.2 Visualisation sur protege (OWL)	20

1.Définition NLP :

Le NLP pour Natural Language Processing ou Traitement Numérique du Langage est une discipline qui porte essentiellement sur la compréhension, la manipulation et la génération du langage naturel par les machines. Ainsi, le *NLP* est réellement à l'interface entre la science informatique et la linguistique. Il porte donc sur la capacité de la machine à interagir directement avec l'humain.

Globalement, nous pouvons distinguer deux aspects essentiels à tout problème de NLP:

- La partie « linguistique », qui consiste à prétraiter et transformer les informations en entrée en un jeu de données exploitable.
- La partie « apprentissage automatique » ou « Data Science », qui porte sur l'application de modèles de Machine Learning ou Deep Learning à ce jeu de données.



Processus de traitement et modélisation des données en NLP

2.Source des données :

Nous avons exporté les données du **PMBOK 5th (41 pages [De la page 309 à 354])** et **PMI's standard (116 pages)** pour former notre 'Corpora' qu'on a ensuite divisé en plusieurs 'Corpus' ou chacun correspond à un processus afin d'organiser les textes et de clarifier les étapes de la gestion du risque. Il s'agit d'un texte non structuré , qui sera par la suite analysé avec les méthodes du prétraitement NLP. ⇒ nous avons utilisé la bibliothèque **pypdf2** pour exporter les textes à partir du pdf .

```
#extraction mel pmi.pdf
import PyPDF2
text_pmi=''
fhandle = open('pmi.pdf', 'rb')
pdfReader = PyPDF2.PdfFileReader(fhandle)
for i in range(12,116):
    pagehandle = pdfReader.getPage(i)
    page=pagehandle.extractText()
    p=str(i-11)
    page=page.replace(p+'@2009 Project Management Institute. ','')|
    page=page.replace('Practice Standard for Project Risk Management','')

    text_pmi=text_pmi + '\n' + page
    text_pmi=text_pmi.lower()
print(text_pmi)
```

```

with open('pmbok_identify_risk.txt', encoding="utf8") as file:
    identify_risk = file.read()

with open('pmbok_plan_risk_management.txt', encoding="utf8") as file:
    plan_risk_management = file.read()

with open('PERFORM_QUALITATIVE_RISK_ANALYSIS.txt', encoding="utf8") as file:
    perform_qualitative_risk_analysis = file.read()

with open('MONITOR AND CONTROL RISKS.txt', encoding="utf8") as file:
    monitor_and_control_risks = file.read()

with open('Plan Risk Responses.txt', encoding="utf8") as file:
    plan_risk_responses = file.read()

with open('PERFORM QUANTITATIVE RISK ANALYSIS.txt', encoding="cp1252") as file:
    perform_quantitative_risk_analysis = file.read()

```

Figure 2 :Importation des fichiers

3.Les phases du prétraitement NLP :

3.1.Nettoyage : cette phase consiste à réaliser des tâches telles que la suppression d'images (les figures), ponctuation.

```

import string

def nettoyage(text):
    text = "".join([char for char in text if char not in string.punctuation])
    text= text.replace(".", "")
    text= text.replace("pmi", "")
    return text
identify_risk=nettoyage(identify_risk)
plan_risk_management=nettoyage(plan_risk_management)
perform_qualitative_risk_analysis=nettoyage(perform_qualitative_risk_analysis)
monitor_and_control_risks=nettoyage(monitor_and_control_risks)
plan_risk_responses=nettoyage(plan_risk_responses)

```

Figure 3.Code relatif au nettoyage des données

3.2.Normalisation et Exploration des données:

- **Tokenisation:** ou découpage du texte en plusieurs pièces appelées *tokens*, chaque token représente un mot, et identifier des mots semble être une tâche relativement simple.
- **Enlever les mots les plus fréquents :** En anglais, on les appelle les « stop words ». Ces mots, bien souvent, n'apportent pas d'information dans les tâches suivantes. Lorsque l'on effectue par exemple une classification par la méthode Tf-IdF, on souhaite limiter la quantité de mots dans les données d'entraînement, Les « stop words » sont établis comme des listes de mots. Ces listes sont généralement disponibles dans une librairie appelée NLTK (Natural Language ToolKit), et dans beaucoup de langues différentes.

- **Lemmatisation** : cela consiste à réaliser la même tâche mais en utilisant un vocabulaire et une analyse fine de la construction des mots. La lemmatisation permet donc de supprimer uniquement les terminaisons inflexibles et donc à isoler la forme canonique du mot, connue sous le nom de lemme. *Exemple* : « *trouvez* » -> *trouvez*.

```
def convert(list):
    return tuple(list)
# tokenize , lem , stop sur text_pmi

def preprocess(text):
    i=0
    stop_words=set(stopwords.words("english"))

    words=[]
    try:

        token = word_tokenize(text)
        words.append(token)

    except KeyError:
        pass

    for w in words:
        my_list = list(w)

        for w1 in w:

            if w1 in stop_words:
                my_list.remove(w1)
        return my_list
```

figure 4 : code relatif à la phase normalisation de données

```
[('identify', 'risk', 'identifies', 'many', 'knowable', 'risk', 'practicable', '51', 'purpose', 'objective', 'identify', 'ris
k', 'process', 'risk', 'managed', 'unless', 'fi', 'rst', 'identified', 'consequently', 'risk', 'management', 'planning', 'h
a', 'completed', 'fi', 'rst', 'process', 'iterative', 'project', 'risk', 'management', 'process', 'aim', 'identify', 'knowabl
e', 'risk', 'project', 'objective', 'however', 'impossible', 'identify', 'risk', 'outset', 'project', 'time', 'level', 'proje
ct', 'risk', 'exposure', 'change', 'result', 'decision', 'action', 'taken', 'previously', 'project', 'internal', 'change', 'e
xternally', 'imposed', 'change', 'purpose', 'risk', 'identifi', 'cation', 'identify', 'risk', 'maximum', 'extent', 'practicab
le', 'fact', 'some', 'risk', 'unknowable', 'emergent', 'requires', 'identify', 'risk', 'process', 'iterative', 'repeating',
'identify', 'risk', 'process', 'fi', 'nd', 'new', 'risk', 'become', 'knowable', 'since', 'previous', 'iteration', 'process',
'risk', 'fi', 'rst', 'identified', 'potential', 'response', 'may', 'also', 'identified', 'time', 'recorded', 'identify', 'ris
k', 'process', 'considered', 'immediate', 'action', 'action', 'appropriate', 'response', 'implemented', 'immediately', 'consi
dered', 'plan', 'risk', 'response', 'process', '52', 'critical', 'success', 'factor', 'identify', 'risk', 'process', 'practic
e', 'described', 'section', '521', '5210', 'maximize', 'value', 'effectiveness', 'identify', 'risk', 'process', 'enhance', 'l
ikelihood', 'identifying', 'many', 'risk', 'practicable', '521', 'early', 'identifi', 'cation', 'risk', 'identifi', 'cation',
'performed', 'early', 'possible', 'project', 'lifecycle', 'recognizing', 'paradox', 'uncertainty', 'high', 'initial', 'stag
e', 'project', 'often', 'le', 'information', 'base', 'risk', 'identifi', 'cation', 'early', 'risk', 'identifi', 'cation', 'en
ables', 'key', 'project', 'decision', 'take', 'maximum', 'account', 'risk', 'inherent', 'project', 'may', 'result', 'change',
'project', 'strategy', 'also', 'maximizes', 'time', 'available', 'development', 'implementation', 'risk', 'response', 'enhanc
es', 'effi', 'ciency', 'since', 'response', 'taken', 'early', 'often', 'normally', 'le', 'costly', 'later', 'one', '522', 'it
erative', 'identifi', 'cation', 'since', 'risk', 'identified', 'given', 'point', 'project', 'essential', 'risk', 'identifi',
```

- **Passage en minuscule:**

```
identify_risk=identify_risk.lower()
plan_risk_management=plan_risk_management.lower()
perform_qualitative_risk_analysis=perform_qualitative_risk_analysis.lower()
monitor_and_control_risks=monitor_and_control_risks.lower()
plan_risk_responses=plan_risk_responses.lower()
perform_quantitative_risk_analysis=perform_quantitative_risk_analysis.lower()
```

Figure 6 : Code relatif passage en minuscule

- **POS tagging:** l'étiquetage morpho-syntaxique est le processus qui consiste à associer aux mots d'un texte les informations grammaticales correspondantes comme la partie du discours, le genre, le nombre, etc.



```
for cmp in retour_identify_risk:
    pos_tag_identify_risk=nltk.pos_tag(cmp)
    print(pos_tag_identify_risk)
for cmp in retour_plan_risk_management:
    pos_tag_plan_risk_management=nltk.pos_tag(cmp)
for cmp in retour_perform_qualitative_risk_analysis:
    pos_tag_perform_qualitative_risk_analysis=nltk.pos_tag(cmp)
for cmp in retour_monitor_and_control_risks:
    pos_tag_monitor_and_control_risks=nltk.pos_tag(cmp)
for cmp in retour_plan_risk_responses:
    pos_tag_retour_plan_risk_responses=nltk.pos_tag(cmp)
for cmp in retour_perform_quantitative_risk_analysis:
    pos_tag_retour_perform_quantitative_risk_analysis=nltk.pos_tag(cmp)
```

```
[('identify', 'NN'), ('risk', 'NN'), ('identifies', 'NNS'), ('many', 'JJ'), ('knowable', 'JJ'), ('risk', 'NN'), ('practicabl', 'JJ'), ('51', 'CD'), ('purpose', 'JJ'), ('objective', 'JJ'), ('identify', 'NN'), ('risk', 'NN'), ('process', 'NN'), ('risk', 'NN'), ('managed', 'VBD'), ('unless', 'IN'), ('fi', 'JJ'), ('rst', 'NN'), ('identified', 'VBN'), ('consequently', 'RB'), ('risk', 'JJ'), ('management', 'NN'), ('planning', 'NN'), ('ha', 'NN'), ('completed', 'VBD'), ('fi', 'JJ'), ('rst', 'JJ'), ('process', 'NN'), ('iterative', 'NN'), ('project', 'NN'), ('risk', 'NN'), ('management', 'NN'), ('process', 'NN'), ('aim', 'NN'), ('identify', 'NN'), ('knowable', 'JJ'), ('risk', 'NN'), ('project', 'NN'), ('objective', 'JJ'), ('however', 'RB'), ('impossible', 'JJ'), ('identify', 'NN'), ('risk', 'NN'), ('outset', 'IN'), ('project', 'NN'), ('time', 'NN'), ('level', 'NN'), ('project', 'NN'), ('risk', 'NN'), ('exposure', 'NN'), ('change', 'NN'), ('result', 'NN'), ('decision', 'NN'), ('action', 'NN'), ('taken', 'VBN'), ('previously', 'RB'), ('project', 'JJ'), ('internal', 'JJ'), ('change', 'NN'), ('externally', 'RB'), ('imposed', 'VBN'), ('change', 'NN'), ('purpose', 'NN'), ('risk', 'NN'), ('identify', 'VBP'), ('cation', 'NN'), ('identify', 'NN'), ('risk', 'NN'), ('maximum', 'JJ'), ('extent', 'NN'), ('practicable', 'JJ'), ('fact', 'NN'), ('some', 'DT'), ('risk', 'NN'), ('unknownable', 'JJ'), ('emergent', 'JJ'), ('requires', 'VBZ'), ('identify', 'JJ'), ('risk', 'NN'), ('process', 'NN'), ('iterative', 'JJ'), ('repeating', 'NN'), ('identify', 'NN'), ('risk', 'NN'), ('process', 'NN'), ('fi', 'NN'), ('nd', 'RB'), ('new', 'JJ'), ('risk', 'NN'), ('become', 'NN'), ('knowable', 'JJ'), ('since', 'IN'), ('previous', 'JJ'), ('iteration', 'NN'), ('process', 'NN'), ('risk', 'NN'), ('fi', 'VBP'), ('rst', 'RB'), ('identified', 'VBN'), ('potential', 'JJ'), ('response', 'NN'), ('may', 'MD'), ('also', 'RB'), ('identified', 'VB'), ('time', 'NN'), ('recorded', 'VBN'), ('identify', 'NN'), ('risk', 'NN'), ('process', 'NN'), ('considered', 'VBN'), ('immediate', 'JJ'), ('action', 'NN'), ('action', 'NN'), ('appropriate', 'JJ'), ('response', 'NN'), ('implemented', 'VBD'), ('immediately', 'RB'), ('considered', 'VBN'), ('plan', 'NN'), ('risk', 'NN'), ('response', 'NN'), ('process', 'NN'), ('52', 'CD'), ('critical', 'JJ'), ('success', 'NN'), ('factor', 'NN'), ('identi
```

Figure 7 : code+résultat pos tagging

- **Recherche et Remplacement des Synonymes (wordnet):**

```
# df must have a column named : Preprocess (with capital P)
def find_synonyms(df):
    list_synonyms=[]
    list_words=[]
    lista=[]

    for sentence in df['Preprocessed']:
        for word in sentence:
            if word not in list_words:
                list_words.append(word)

    for word in list_words:
        syns = wordnet.synsets(word)
        for syn in syns:
            for l in syn.lemmas():
                if l.name() not in lista:
                    lista.append(l.name())
    list_synonyms.append(lista)
    lista=[]

    return list_words, list_synonyms
```

```

# df must have a column named : Preprocess (with capital P)
def find_synonyms(df):
    list_synonyms=[]
    list_words=[]
    lista=[]

    for sentence in df['Preprocessed']:
        for word in sentence:
            if word not in list_words:
                list_words.append(word)
    for word in list_words:
        syns = wordnet.synsets(word)
        for syn in syns:
            for l in syn.lemmas():
                if l.name() not in lista:
                    lista.append(l.name())
        list_synonyms.append(lista)
        lista=[]

    return list_words, list_synonyms

```

Figure 9 : code relatif à la recherche et au remplacement des synonymes

- **Chunking:** c'est un processus d'extraction de phrases à partir de texte non structuré, ce qui signifie analyser une phrase pour identifier les constituants (groupes de noms, verbes, groupes de verbes, etc.) Il utilise des balises POS en entrée et fournit des morceaux en sortie. En bref, Chunking signifie regrouper des mots/jetons en morceaux

```

nltk.download('averaged_perceptron_tagger')
for cmp in retour_identify_risk:

    chunkGram = r"""Chunk: {<RB.?*>*<VB.?*>*<NNP>+<NN>?}"""
    chunkParser = nltk.RegexpParser(chunkGram)
    chunked = chunkParser.parse(pos_tag)
    chunked.draw()

```

- **Spacy Rule Matching:** Le Matcher nous permet de trouver des mots et des phrases à l'aide de règles décrivant leurs attributs de jeton. L'application du matcher à un Doc nous donne accès aux mots composés(bi/tri-gram).

```

matcher = Matcher(nlp.vocab)
patterns = [
    [{'POS': 'NOUN'}, {'POS': 'VERB'}, {'POS': 'NOUN'}],
    [{'POS': 'NOUN'}, {'POS': 'NOUN'}],
    [{'POS': 'NOUN'}, {"POS": "ADJ"}, {'POS': 'NOUN'}],
    [{'POS': 'NOUN'}, {"IS_SPACE": True}, {'POS': 'NOUN'}],
    [{'POS': 'NOUN'}, {"IS_STOP": True}, {'POS': 'NOUN'}]
]
matches_by_document=[]
for document in corpus:
    doc = nlp(document)
    matches = matcher(doc)
    ithem=set()
    for pattern in patterns:
        matcher.add("id1", [pattern])
        # We will show you the first 20 matches
        ithem.update(set([doc[start:end].text for match_id, start, end in matches]))
        #print("Matches:", [doc[start:end].text for match_id, start, end in matches])
    matches_by_document.append(ithem)
matches_by_document

```


4.TF-IDF

Le TF-IDF (de l'anglais term frequency-inverse document frequency) est une méthode de pondération souvent utilisée en recherche d'information et en particulier dans la fouille de textes. cette mesure statistique permet d'évaluer l'importance d'un terme contenu dans un document. Nous allons donc calculer la valeur de TF-IDF des termes de base (token a un seul mot) et des termes obtenus à partir du bi-gram et du n-gram.

```
corpus_by_token_bi_tri_gram=[]
unique_gram=[]
for doc in corpus:
    token = nltk.word_tokenize(doc)
    bigram = bi_gram(token)
    triam = tri_gram(token)
    All = token + bigram+triam
    unique_gram=unique_gram+All
    corpus_by_token_bi_tri_gram.append(list(set(All)))
corpus_by_token_bi_tri_gram
list(set(unique_gram))
```

Figure 12 : code relatif au regroupement de tous les termes

- **Term frequency (TF):** Le nombre de fois qu'un terme apparaît dans un document divisé par le nombre total de mots dans le document.

```
def numOfWords(x,unique_words):
    numOfWords = dict.fromkeys(unique_words, 0)
    for word in x:
        numOfWords[word]+= 1
    return numOfWords
def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
    return tfDict
def computeIDF(documents):
```

Figure 13: code relatif au calcul de TF

- **Inverse Data Frequency (IDF):** Le log de nombre de documents divisé par le nombre de documents contenant le mot W. La fréquence inverse des données détermine le poids des mots rares dans tous les documents du corpora.

$$idf(w) = \log\left(\frac{N}{df_t}\right)$$

```
def computeIDF(documents):
    import math
    N = len(documents)

    idfDict = dict.fromkeys(documents[0].keys(), 0)
    for document in documents:
        for word, val in document.items():
            if val > 0:
                idfDict[word] += 1

    for word, val in idfDict.items():
        idfDict[word] = math.log(N / float(val))
    return idfDict
```

Figure 14 : code relatif au calcul de IDF

- **TF-IDF:** C'est simplement TF multiplié par IDF.

```
def computeTFIDF(tfBagOfWords, idfs):
    tfidf = {}
    for word, val in tfBagOfWords.items():
        tfidf[word] = val * idfs[word]
    return tfidf

numOfWords_corpus=[]
computeTF_corpus=[]
for i in corpus_by_token_bi_tri_gram:
    num=numOfWords(i,unique_gram)
    numOfWords_corpus.append(num)
    compTF=computeTF(num,i)
    computeTF_corpus.append(compTF)
idfs=computeIDF(numOfWords_corpus)
TFIDF_corpus=[]
for i in computeTF_corpus:
    TFIDF_corpus.append(computeTFIDF(i, idfs))
df = pd.DataFrame(TFIDF_corpus)
print(df)
select_key_words_by_percent(10)
```

5. Named Entities Extraction:

La reconnaissance d'entités nommées est une sous-tâche d'extraction d'informations qui cherche à localiser et à classer les entités nommées mentionnées dans un texte non structuré en catégories prédéfinies telles que les noms de personnes, les organisations, les emplacements, les codes médicaux, les expressions temporelles, les quantités, les valeurs monétaires, les pourcentages, etc. .

```
def entityExtraction(s):
    doc = nlp(s)
    d = []
    for ent in doc.ents:
        d.append((ent.label_, ent.text))
    df_entity = pd.DataFrame(d, columns = ('named entity', 'output'))
    return df_entity
df_raw_text_entity = entityExtraction(combined_processes)
df_raw_text_entity
```

	index	named entity	output
0	0	ORG	WBS
1	3	ORG	Project Risk Management
2	7	ORG	Google
3	10	ORG	the Project Risk Management
4	14	ORG	Identify Risks
5	18	ORG	Delphi
6	19	ORG	Delphi
7	21	ORG	RBS
8	23	ORG	SWOT
9	24	ORG	SWOT

Figure 17 : Résultat de l'extraction des entités nommées

6. Cosine Similarity Measure:

La distance de Levenshtein entre mots ou chaînes de caractères (distance d'édition, de similarité) donne par un calcul assez simple des indications sur le degré de ressemblance de ces chaînes. C'est le nombre minimum des changements de caractères effectué à une séquence afin d'obtenir une autre.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

sentences = list(['yessine is a web developer', 'wassim is a data scientist'])
vectorizer = CountVectorizer().fit_transform(sentences)
vectors = vectorizer.toarray()
```

vectors

```
array([[0, 1, 1, 0, 0, 1, 1],
       [1, 0, 1, 1, 1, 0, 0]], dtype=int64)
```

```
def cosine_two_vectors(vec1, vec2):
    vec1 = vec1.reshape(1,-1)
    vec2 = vec2.reshape(1,-1)
    return cosine_similarity(vec1, vec2)[0][0]
```

```
cosine_two_vectors(vectors[0], vectors[1])
```

0.25

7. Latent Dirichlet Allocation:

LDA est utilisé pour classer le texte dans un document à un sujet particulier. Il construit un sujet par modèle de document et des mots par modèle de sujet, modélisés comme des distributions de Dirichlet.

```
def format_topics_sentences(ldamodel=None, corpus=corpora, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(ldamodel[corpora]):
        row = row_list[0] if ldamodel.per_word_topics else row_list
        # print(row)
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0: # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4), topic_keywords]), ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)

df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model, corpus=corpora, texts=data_words)

# Format
df_dominant_topic = df_topic_sents_keywords.reset_index()
df_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'Keywords', 'Text']
df_dominant_topic.head(10)
```

Figure 19 : code relatif a la phase de LDA

Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	6.0	0.6999 risks, planning, analysis, process, identifies...	[identifies, risks]
1	1	3.0	0.7749 risks, taking, training, planning, making, pro...	[managing, identifies, risks]
2	2	5.0	0.8875 risks, planning, training, process, responses,...	[management, risks, complete, planning, identi...
3	3	6.0	0.8200 risks, planning, analysis, process, identifies...	[planning, identifies, outset, risks]
4	4	6.0	0.9250 risks, planning, analysis, process, identifies...	[leveling, training, process, impose, transfer...
5	5	5.0	0.8499 risks, planning, training, process, responses,...	[risks, extent, identification, identifies, tr...
6	6	3.0	0.9000 risks, taking, training, planning, making, pro...	[become, iteration, fact, risks, repeat, findi...
7	7	6.0	0.8199 risks, planning, analysis, process, identifies...	[identifies, risks, timing, responses]
8	8	7.0	0.7749 risks, process, planning, responses, training,...	[record, process, thinking]
9	9	5.0	0.8500 risks, planning, training, process, responses,...	[implementing, risks, responses, process, thin...

Figure 20: 1ér Résultat de la phase de LDA

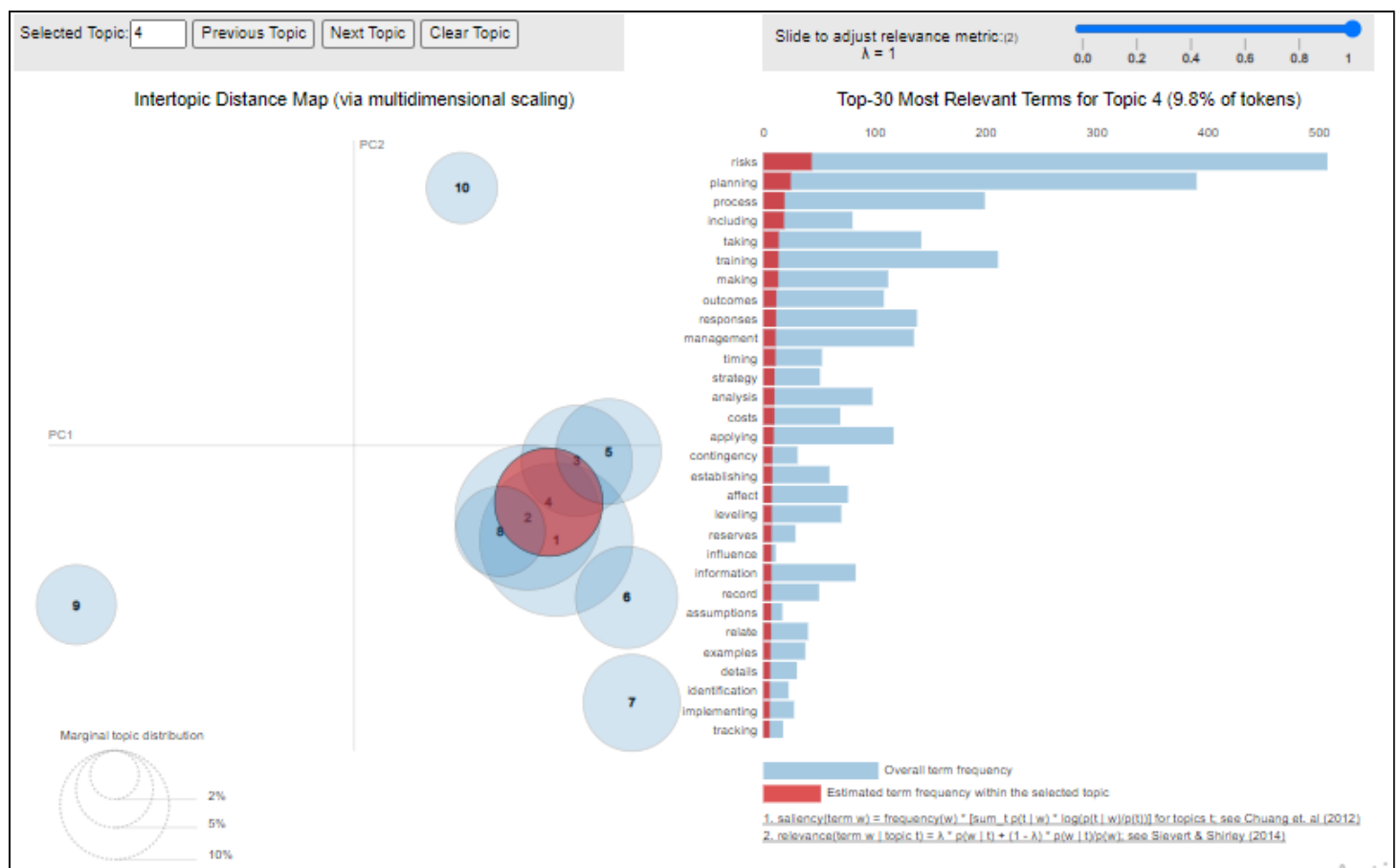


Figure 21 : 2ème Résultat de la phase de LDA

8. Extraction des relations:

Nous avons opté à la méthode spacy matching du bibliothèque spacy ou nous avons défini les différents patterns afin d'extraire des relations par processus , ainsi nous avons pris en considération les mots composés pour conserver leur sens technique .

```
def relation_1(corpus):
    matcher_1 = Matcher(nlp.vocab)
    patterns_1 = [

        [{ 'POS': 'NOUN'}, { 'POS': 'VERB'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'VERB'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'VERB'}, { 'POS': 'NOUN'}],

        [{ 'POS': 'NOUN'}, { 'POS': 'AUX'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'AUX'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'AUX'}, { 'POS': 'NOUN'}],

    ]

    doc = nlp(corpus)
    matcher_1.add("id1", patterns_1)
    matches_1 = matcher_1(doc)
    matches_1 = set([doc[start:end].text for match_id, start, end in matches_1])
    matcher_1.remove("id1")
    matches_1

def relation_2(corpus):
    matcher_2 = Matcher(nlp.vocab)
    patterns_2 = [

        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'VERB'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'VERB'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'VERB'}, { 'POS': 'NOUN'}],

        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'AUX'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'AUX'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'AUX'}, { 'POS': 'NOUN'}],

    ]

def relation_3(corpus):
    matcher_3 = Matcher(nlp.vocab)
    patterns_3 = [

        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'VERB'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'VERB'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'VERB'}, { 'POS': 'NOUN'}],

        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'AUX'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'AUX'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}],
        [{ 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'NOUN'}, { 'POS': 'AUX'}, { 'POS': 'NOUN'}],

    ]
```

Figure 22: Code des relations(spacy matching)

relations_by_process[1]

	Relation	Lefts	Predicate	Rights	Process
0	member remain importance	member	remain	importance	monitor_and_control_risks
1	reserves costs planning	reserves	cost	planning	monitor_and_control_risks
2	milestone facilitate forecasting	milestone	facilitate	forecasting	monitor_and_control_risks
3	outcomes change risks	outcomes	change	risks	monitor_and_control_risks
4	decision add case	decision	add	case	monitor_and_control_risks
...
235	team member remain importance	team_member	remain	importance	monitor_and_control_risks
236	action owner take costs	action_owner	take	costs	monitor_and_control_risks
237	action owner controlling status	action_owner	control	status	monitor_and_control_risks
238	variance analysis compare outcomes	variance_analysis	compare	outcomes	monitor_and_control_risks
239	management planning include risks	management_planning	include	risks	monitor_and_control_risks

240 rows × 5 columns

relations_by_process[2]

	Relation	Lefts	Predicate	Rights	Process
0	source cause group	source	cause	group	perform_qualitative_risk_analysis
1	activity risks management	activity	risk	management	perform_qualitative_risk_analysis
2	updates updates risks	updates	update	risks	perform_qualitative_risk_analysis
3	risks be risks	risks	be	risks	perform_qualitative_risk_analysis
4	input be tailor	input	be	tailor	perform_qualitative_risk_analysis
...
234	analysis process include use	analysis_process	include	use	perform_qualitative_risk_analysis
235	impact matrix contribute risks	impact_matrix	contribute	risks	perform_qualitative_risk_analysis
236	watch list contribute contingency	watch_list	contribute	contingency	perform_qualitative_risk_analysis
237	analysis process contribute structure	analysis_process	contribute	structure	perform_qualitative_risk_analysis
238	schedule activity risks management	schedule_activity	risk	management	perform_qualitative_risk_analysis

239 rows × 5 columns

relations_by_process[3]					
	Relation	Lefts	Predicate	Rights	Process
0	tools permit combination	tools	permit	combination	perform_quantitative_risk_analysis
1	costs compare plan	costs	compare	plan	perform_quantitative_risk_analysis
2	chart applying sensitivity analysis	chart	apply	sensitivity_analysis	perform_quantitative_risk_analysis
3	bias making reference	bias	make	reference	perform_quantitative_risk_analysis
4	method plan scheduling	method	plan	scheduling	perform_quantitative_risk_analysis
...
277	tornado diagram calculating costs	tornado_diagram	calculate	costs	perform_quantitative_risk_analysis
278	update could including analysis	update_could	include	analysis	perform_quantitative_risk_analysis
279	sensitivity analysis costs tornado	sensitivity_analysis	cost	tornado	perform_quantitative_risk_analysis
280	tree analysis incorporate risks	tree_analysis	incorporate	risks	perform_quantitative_risk_analysis
281	sensitivity analysis help determine risks	sensitivity_analysis_help	determine	risks	perform_quantitative_risk_analysis
282 rows × 5 columns					

relations_by_process[5]					
	Relation	Lefts	Predicate	Rights	Process
0	attitude adapt risks	attitude	adapt	risks	plan_risk_responses
1	roles thresholds definition	roles	thresholds	definition	plan_risk_responses
2	process be document	process	be	document	plan_risk_responses
3	activity risks management	activity	risk	management	plan_risk_responses
4	risks affect type	risks	affect	type	plan_risk_responses
...
277	management context be combination	management_context	be	combination	plan_risk_responses
278	planning meeting risks management activity	planning_meeting	risk	management_activity	plan_risk_responses
279	management activity risks management	management_activity	risk	management	plan_risk_responses
280	management activity ensure organization	management_activity	ensure	organization	plan_risk_responses
281	schedule contingency reserve making risks	schedule_contingency_reserve	make	risks	plan_risk_responses
282 rows × 5 columns					

relations_by_process[4]					
	Relation	Lefts	Predicate	Rights	Process
0	tools permit combination	tools	permit	combination	plan_risk_management
1	costs compare plan	costs	compare	plan	plan_risk_management
2	chart applying sensitivity analysis	chart	apply	sensitivity_analysis	plan_risk_management
3	bias making reference	bias	make	reference	plan_risk_management
4	method plan scheduling	method	plan	scheduling	plan_risk_management
...
277	tornado diagram calculating costs	tornado_diagram	calculate	costs	plan_risk_management
278	update could including analysis	update_could	include	analysis	plan_risk_management
279	sensitivity analysis costs tornado	sensitivity_analysis	cost	tornado	plan_risk_management
280	tree analysis incorporate risks	tree_analysis	incorporate	risks	plan_risk_management
281	sensitivity analysis help determine risks	sensitivity_analysis_help	determine	risks	plan_risk_management
282 rows × 5 columns					

Figure 23: Les relations dégagés par process avec spacy matching

8.1.Extraction des classes :

Les noms obtenu de l'extraction des relations représentent les concepts , nous avons concaténer ces derniers afin d'obtenir les classes :

Classes Extraction

```
##### Extracting Classes #####

list_classes_by_process = []
for name in process_names:
    ##### Extracting classes from the left side of the relations #####

    left_classes = project_risk_management_relations.loc[project_risk_management_relations["Process"] == name].copy()
    left_classes.reset_index(inplace=True, drop=True)
    copy = left_classes.copy()
    for i in range(len(copy)):
        copy.iloc[i]['Lefts'] = copy.iloc[i]['Lefts'].strip()
        if " " in copy['Lefts'].iloc[i]:
            left_classes.drop(i, axis=0, inplace = True)
    left_classes = left_classes['Lefts'].values

    ##### Extracting classes from the right side of the relations #####

    right_classes = project_risk_management_relations.loc[project_risk_management_relations["Process"] == name].copy()
    right_classes.reset_index(inplace=True, drop=True)
    copy = right_classes.copy()
    for i in range(len(copy)):
        copy.iloc[i]['Rights'] = copy.iloc[i]['Rights'].strip()
        if " " in copy['Rights'].iloc[i]:
            right_classes.drop(i, axis=0, inplace = True)
```

Activer Windows
Accédez aux paramètres pour a

Figure 24: Extraction des classes par process

8.2.Extraction des subclasses :

Sub-classes Extraction

```
for k in range(len(classes_by_process)):
    subclasses = []
    df = relations_by_process[k].copy()
    for c in classes_by_process[k]['Classes']:
        left_subclasses = []
        for i in range(len(df)):
            term = df.iloc[i]['Lefts'].strip()
            # if c in term and term != c and not(term.startswith(c)):
            if term.endswith((" "+c)) and not(term.startswith(c)):
                left_subclasses.append(term)

        right_subclasses = []
        for i in range(len(df)):
            term = df.iloc[i]['Rights'].strip()
            # if c in term and term != c and not(term.startswith(c)):
            if term.endswith((" "+c)) and not(term.startswith(c)):
                right_subclasses.append(term)

        subclasses.append(list(set(list(left_subclasses)+list(right_subclasses))))
    classes_by_process[k]['Subclasses'] = subclasses
    classes_by_process[k]['Process'] = process_names[k]
```

Figure 25: Extraction des subclasses par process

	Classes	Process	Subclasses
0	expert	identify_risk	[]
1	language	identify_risk	[]
2	iteration	identify_risk	[]
3	assumptions	identify_risk	[]
4	provision	identify_risk	[]
...
682	training	plan_risk_responses	[]
683	credibility	plan_risk_responses	[]
684	size	plan_risk_responses	[]
685	skill	plan_risk_responses	[]
686	asset	plan_risk_responses	[process_asset,]
687 rows x 3 columns			

Figure 26: Output des classes et subclasses par process

9. Ontology learning process:

En informatique et en science de l'information, 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.

Parallèlement à cette définition assez théorique de ce que représente une ontologie, une autre définition, plus opérationnelle, peut être formulée ainsi :

« Une ontologie est un réseau sémantique qui regroupe un ensemble de concepts décrivant complètement un domaine. Ces concepts sont liés les uns aux autres par des relations taxinomiques (hiérarchisation des concepts) d'une part, et sémantiques d'autre part. »



9.1 Processus de construction de composants OWL

9.1.1.RDF(Resource Description Framework) :

Un document structuré en RDF est un ensemble de triplets.

Un triplet RDF est une association (*sujet, prédicat, objet*) :

- le « sujet » représente la ressource à décrire ;
- le « prédicat » représente un type de propriété applicable à cette ressource ;
- l' « objet » représente une donnée ou une autre ressource : c'est la valeur de la propriété.

Le sujet et l'objet, dans le cas où ce sont des ressources, peuvent être identifiés par un URI ou être des nœuds anonymes. Le prédicat est nécessairement identifié par un URI.

Les documents RDF peuvent être écrits en différentes syntaxes, y compris en XML. Mais RDF en soi n'est pas un dialecte XML. Il est possible d'avoir recours à d'autres syntaxes pour exprimer les triplets. RDF est simplement une structure de données constituée de nœuds et organisée en graphe. Bien que RDF/XML ne soit qu'une syntaxe (ou sérialisation) du modèle, elle est souvent appelée RDF, par abus de langage.

Un document RDF ainsi formé correspond à un multigraphe orienté étiqueté. Chaque triplet correspond alors à une arête orientée dont l'étiquette est le prédicat, le nœud source est le sujet et le nœud cible est l'objet.

⇒ En utilisant **RDFLIB** , nous avons instancié le graphe g puis nous avons ajouté les classes , les subclasses , les individuals et les objects properties , ensuite nous avons ajouté des préfixes aux entités afin de les différencier et puis nous avons extrait le graphe sous format .nt pour le visualiser dans le logiciel protégé

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

<Graph identifier=N04c410b792a64c1a8a290f722ab723c7 (<class 'rdflib.graph.Graph'>>
```

Figure 27: Construction du graphe RDF

9.1.2. : Ajout des classes

Nous avons ajouté les classes au graphe RDF :

```
for process in process_names:
    p = URIRef('http://www.example.org/' + process)
    uri = 'http://www.example.org/' + process + "/"
    g.add((p, RDF.type, OWL.Class))
```

Figure 28: Ajout des classes au graphe

9.1.3. : Ajout des subclasses

Nous avons ajouté des subclasses au graphe RDF :

```
for c in final_project_risk_management_classes['Classes'].loc[final_project_risk_management_classes['Process'] == process]:
    cl = URIRef(uri + c)
    g.add((cl, RDFS.subClassOf, p))

for all_sc in final_project_risk_management_classes['Subclasses'].loc[final_project_risk_management_classes['Classes'] == process]:
    for sc in all_sc:
        if sc:
```

Figure 30: Ajout des subclasses au graphe

9.1.4. : Ajout des individuals

Nous avons ajouté les individuals :

```
for all_indivs in final_project_risk_management_classes['Individuals'].loc[final_project_risk_management_classes['Classe'] == process]:
    for indiv in all_indivs:
        if indiv:
            individ = URIRef(str(uri)+str(indiv))
            g.add((individ, RDF.type, cl))
for all_indivs in final_project_risk_management_classes['Individuals'].loc[final_project_risk_management_classes['Classe'] == process]:
    for indiv in all_indivs:
        if indiv:
            individ = URIRef(str(uri)+str(indiv))
            g.add((individ, RDF.type, OWL.NamedIndividual))
```

Activer Windows

Figure 31: Ajout des individuals au graphe

9.1.5. : Ajout des object properties

Nous avons ajouté les object properties :

```
for i in range(len(project_risk_management_relations)):
    relation = project_risk_management_relations.iloc[i]
    uri = 'http://www.example.org/' + relation['Process'] + "/"
    prop = URIRef(uri + relation['Predicate'])
    subj = URIRef(uri + relation['Lefts'])
    obj = URIRef(uri + relation['Rights'])
    g.add((prop, RDF.type, OWL.ObjectProperty))
    g.add((prop, RDFS.domain, subj))
    g.add((prop, RDFS.range, obj))
```

Figure 32: Ajout des object properties

```
g.serialize(destination='RDF.nt', format='nt')
```

```
<Graph identifier=N04c410b792a64c1a8a290f722ab723c7 (<class 'rdflib.graph.Graph'>>>
```

Figure 32: Ajout des object properties et exportation du graphe

9.2 Visualisation sur protege (OWL)

D'abord nous avons exporté le graphe .nt puis nous avons converti ce dernier en un fichier .OWL

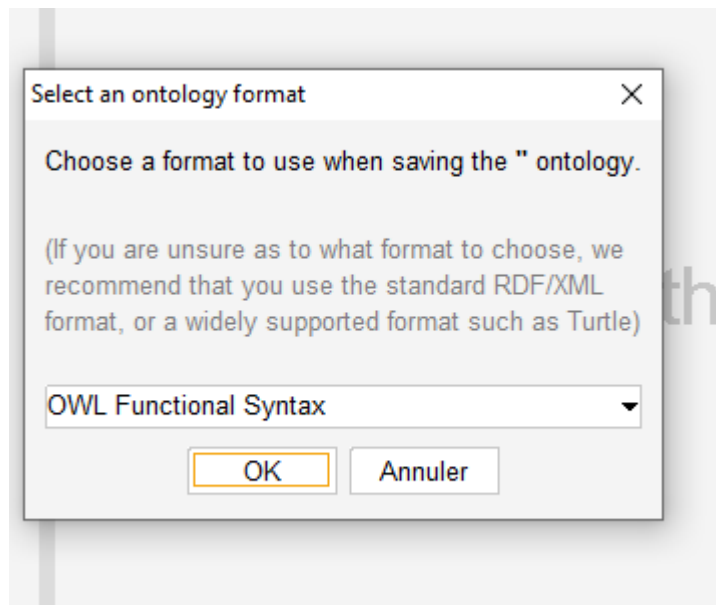
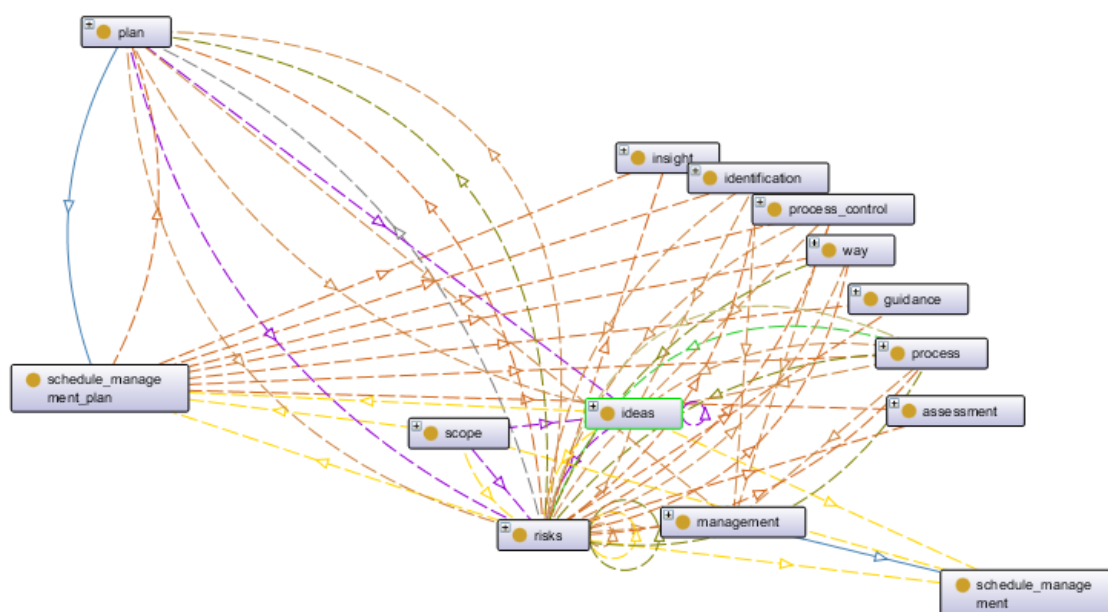
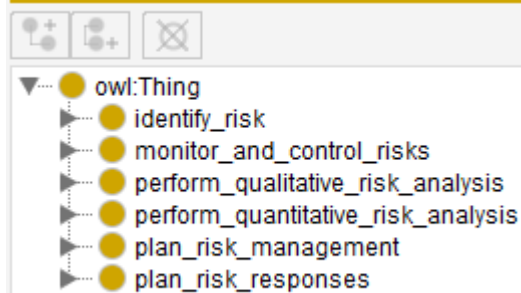


Figure 33: Conversion du .nt en OWL

Class hierarchy:



Usage: analyze

Show: ☒ this ☒ disjoints

Found 9 uses of analyze

- analyze
 - Class: analyze
 - analyze SubClassOf monitor_and_control_risks
- analyze
 - analyze Domain audit
 - analyze Range document
 - ObjectProperty: analyze
- take
 - take Range analyze

Figure 34: object properties

- owl:topObjectProperty
 - accept
 - accommodate
 - achieve
 - achieve
 - achieve
 - adapt
 - add
 - add
 - affect
 - agree
 - agree
 - allocate
 - analyze
 - appear
 - apply
 - apply
 - apply
 - approve
 - approve
 - approve
 - assess
 - assess
 - assian

Usage: matrix

Show: ☒ this ☒ different

Found 8 uses of matrix

- affect
 - affect Range matrix
- is_a
 - is_a Domain matrix

Description: matrix ? || ≡ □ ×

Types +

● grid ? @ × ○

Same Individual As +

Different Individuals +

Figure 35: individuals

Individuals: input

◆ ✕

- input
- judgment
- list
- list
- management
- management
- management_activity
- management_context
- management_planning
- management_process
- management_process
- matrix
- mitigation
- mitigation_strategy
- objective
- objective
- opportunity
- opportunity
- organization
- participant
- people
- plan
- planning
- planning
- process
- process

9.3 Les règles d'inférence

Nous avons préparé **18 règles** d'inférence sous le format **SWRL** (*Semantic Web Rule Language*) qui est un langage qui permet d'intégrer des règles d'inférence dans les ontologies. Les règles peuvent être écrites dans l'éditeur Protégé, à l'aide d'Owlready, et ensuite exécutées via les raisonneurs Hermit.

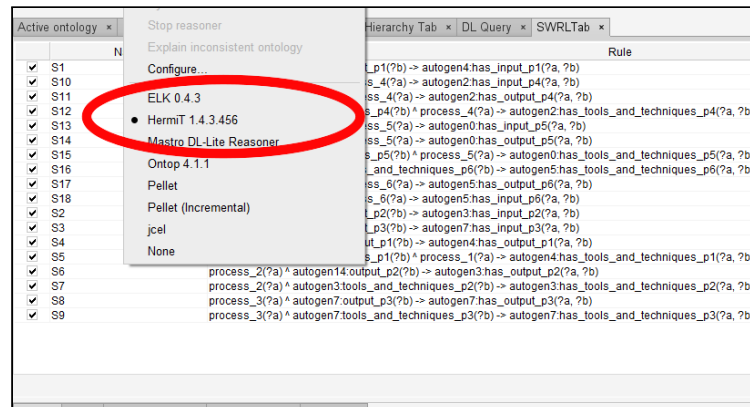


Figure 35: Le raisonneur HERMIT

Une règle SWRL comprend une ou plusieurs conditions et une ou plusieurs conséquences, séparées par une flèche « -> » (caractères moins et plus grand que). Si la règle possède plusieurs conditions ou conséquences, elles sont séparées entre elles par une virgule « , » qui a le sens d'un « et » logique. Les éléments qui composent les conditions et les conséquences sont appelés atomes.

De plus, les règles SWRL utilisent des variables, dont les noms commencent par « ? », par exemple « ?x ». Ces variables peuvent représenter...

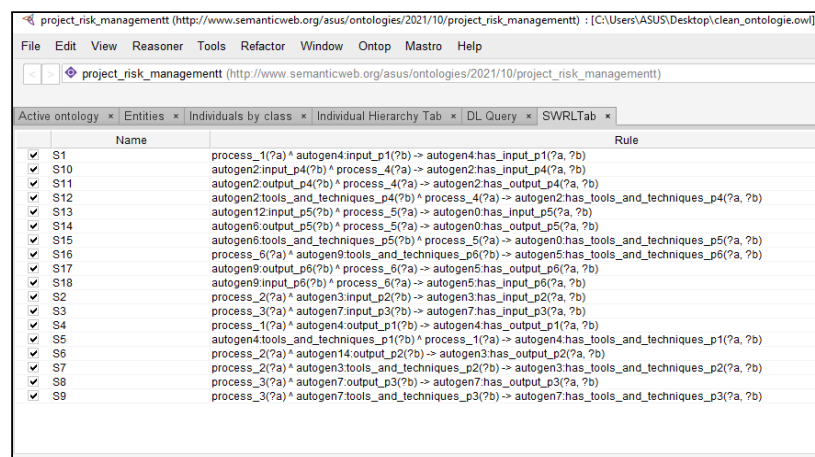
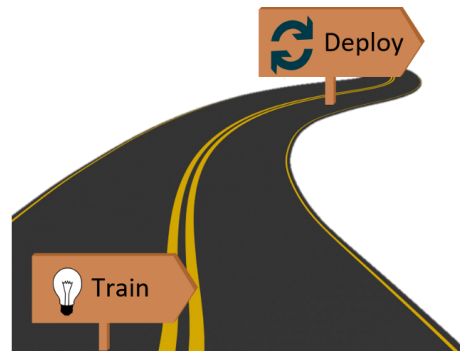


Figure 37: Les 18 règles d'inférence

10.Déploiement :

Il s'agit de l'étape finale du processus. Elle consiste en une mise en production pour les utilisateurs finaux des modèles obtenus. Son objectif : mettre la connaissance obtenue par la modélisation, dans une forme adaptée, et l'intégrer au processus du système de recommandation . Le déploiement peut ainsi aller, selon les objectifs, de la simple génération d'un rapport décrivant les connaissances obtenues jusqu'à la mise en place d'une application, permettant l'utilisation d'un système de recommandation..



10.1 Le logiciel de déploiement:

Django est un cadre de développement web open source en Python. Il a pour but de rendre le développement web 2.0 simple et rapide. Pour cette raison, le projet a pour slogan « Le framework pour les perfectionnistes avec des deadlines. ». Développé en 2003 pour le journal local de Lawrence (État du Kansas, aux États-Unis), Django a été publié sous licence BSD à partir de juillet 2005, Django est un cadre de développement qui s'inspire du principe MVC ou MTV (la vue est gérée par un gabarit) .



10.2 Réalisation du projet :

Type a Keyword

Q




risk

SEARCH

RM Assist

≡ | Q Search Key Words...

View Results



Best match: **identify risk**

Here are your results

identify risk :

Process

Identify Risks is the process of determining which risks may affect the project and documenting their characteristics. The key benefit of this process is the documentation of existing risks and the knowledge and ability it provides to the project team to anticipate events. Participants in risk identification activities may include the following: project manager, project team members, risk management team (if assigned), customers, subject matter experts from outside the project team, end users, other project managers, stakeholders, and risk management experts. While these personnel are often key participants for risk identification, all project personnel should be encouraged to identify potential risks. Identify risks is an iterative process, because new risks may evolve or become known as the project progresses through its life cycle. The frequency of iteration and participation in each cycle will vary by situation. The format of the risk statements should be consistent to ensure that each risk is understood clearly and unambiguously in order to support effective analysis and response development. The risk statement should support the ability to compare the relative effect of one risk against others on the project. The process should involve the project team so they can develop and maintain a sense of ownership and responsibility for the risks and associated risk response actions. Stakeholders outside the project team may provide additional objective information.

View Details

plan risk responses :

Process

Plan Risk Responses is the process of developing options and actions to enhance opportunities and to reduce threats to project objectives. The key benefit of this process is that it addresses the risks by their priority, inserting resources and activities into the budget, schedule and project management plan as needed. The inputs, tools and techniques, and outputs of this process are depicted in Figure 11-18. Figure 11-19 depicts the data flow diagram of the process. The Plan Risk Responses process follows the Perform Quantitative Risk Analysis process (if used). Each risk response requires an understanding of the mechanism by which it will address the risk. This is the mechanism used to analyze if the risk response plan is having the desired effect. It includes the identification and assignment of one person (an owner for risk response) to take responsibility for each agreed-to and funded risk response. Risk responses should be appropriate for the significance of the risk, cost-effective in meeting the challenge, realistic within the project context, agreed upon by all parties involved, and owned by a responsible person. Selecting the optimum risk response from several options is often required. The Plan Risk Responses process presents commonly used approaches to planning responses to the risks. Risks include threats and opportunities that can affect project success, and responses are discussed for each.

View Details

Risk Management Plan :

Input of identify_risk

Here are your results

Identify Risk

Identify Risks is the process of determining which risks may affect the project and documenting their characteristics. The key benefit of this process is the documentation of existing risks and the knowledge and ability it provides to the project team to anticipate events. Participants in risk identification activities may include the following: project manager, project team members, risk management team (if assigned), customers, subject matter experts from outside the project team, end users, other project managers, stakeholders, and risk management experts. While these personnel are often key participants for risk identification, all project personnel should be encouraged to identify potential risks. Identify risks is an iterative process, because new risks may evolve or become known as the project progresses through its life cycle. The frequency of iteration and participation in each cycle will vary by situation. The format of the risk statements should be consistent to ensure that each risk is understood clearly and unambiguously in order to support effective analysis and response development. The risk statement should support the ability to compare the relative effect of one risk against others on the project. The process should involve the project team so they can develop and maintain a sense of ownership and responsibility for the risks and associated risk response actions. Stakeholders outside the project team may provide additional objective information.

Inputs

- Cost_Management_Plan
- Procurement_Documents
- Activity_Cost_Estimates
- Activity_Duration_Estimates

Outputs

- list_of_identified_risks
- list_of_potential_responses
- risk_register

Tools And Techniques

- assumptions_analysis
- brainstorming
- cause_and_effect_diagrams
- checklist_analysis

- Human_Resource_Management_Plan
- Project_Documents
- Risk_Management_Plan
- Schedule_Management_Plan
- Scope_Baseline
- Stakeholder_Register
- enterprise_environmental_factors
- organizational_process_assets
- quality_management_plan

- delphi_technique
- diagramming_techniques
- documentation_reviews
- expert_judgment
- influence_diagrams
- information_gathering_technique
- interviewing
- root_cause_analysis
- swot_analysis
- system_or_process_flow_charts

RM Assist



Q Search Key Words..

View Results



Best match: [Scope Baseline](#)

Here are your results

Scope Baseline :

Input of identify_risk

Described in Section 5.4.3.1. Project assumptions are found in the project scope statement. Uncertainty in project assumptions should be evaluated as potential causes of project risk. The WBS is a critical input to identifying risks as it facilitates an understanding of the potential risks at both the micro and macro levels. Risks can be identified and subsequently tracked at summary, control account, and/or work package levels

scope baseline :

Input of perform_qualitative_risk_analysis

Described in Section 5.4.3.1. Projects of a common or recurrent type tend to have more well-understood risks. Projects using state-of-the-art or first-of-its-kind technology, and highly complex projects, tend to have more uncertainty. This can be evaluated by examining the scope baseline.

Scope baseline :

output of Plan_Risk_Responses

Because of new, modified or omitted work generated by the risk responses, the scope baseline may be updated to reflect those changes.

11.Conclusion et perspectives :

Ce projet vise à fournir un système de recommandation concernant la gestion de risque et ses étapes afin d'offrir plus d'information sur cette process qui est très important dans tous les domaines pour éviter les échecs

Pour cela, nous avons utilisé une méthodologie telle que décrite en deux grandes phases:

phase 1:

1. L'extraction automatique des connaissances (nom(concepts), verbes (relations), les instances de chaque concept, les synonymes,etc) à partir d'un texte de corpus non structuré c'est PMBOK des bonnes pratiques pour la gestion de projet
2. Ces connaissances récupérées seront analysées et filtrés pour les transformer en OWL classes, OWL sub-classes, individuels (instances pour chaque classe), axioms, Object properties et data properties pour modéliser l'ontologie du domaine "PRM Ontology" sous le format OWL file .

phase 2:

1. La construction de règles d'inférence en termes de recommandations sous le format SWRL rules à travers Rule-Based Reasoning process afin de créer la «Base de règles» a partir d'un corpus de recommandation .
2. Ces règles seront déduites pour suggérer les recommandations appropriées à travers un système d'aide à la décision "une application Web".

Nous avons été confrontés à de nombreux défis lors de l'exécution de nos tâches, telles que des difficultés à extraire les données. Cependant, ce seul nous a permis d'acquérir et de développer de nouvelles compétences.

