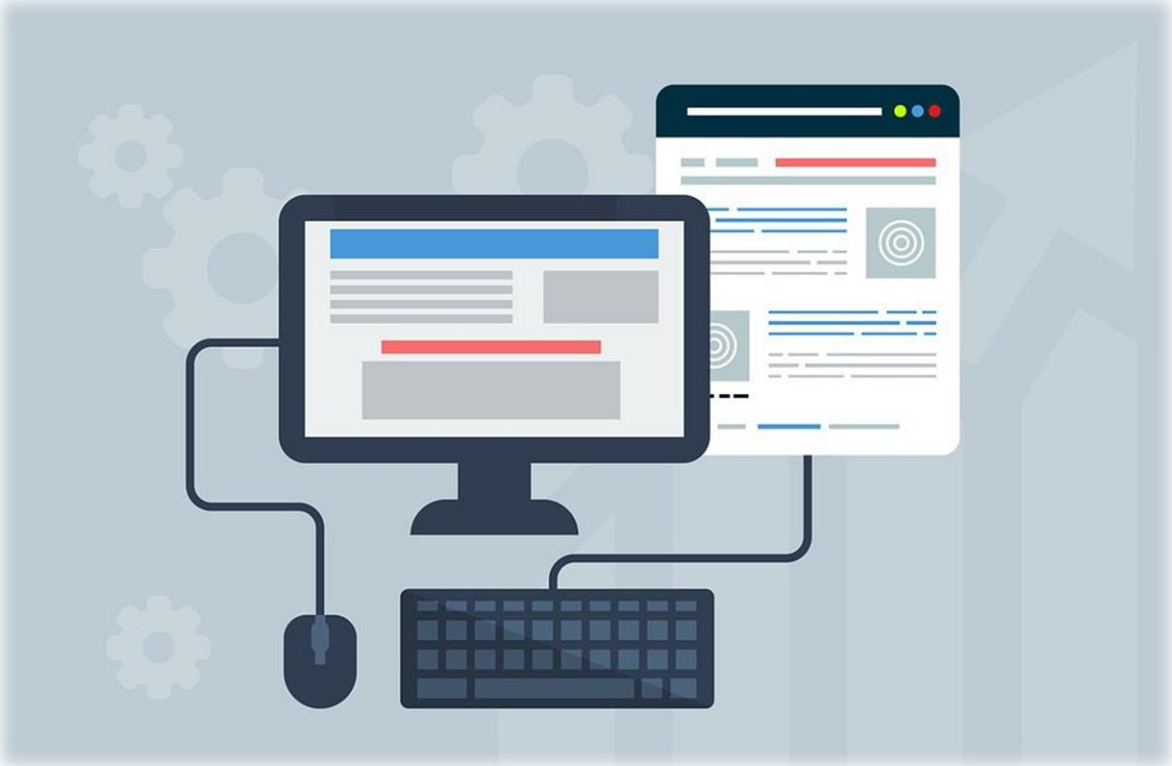
Rapport Projet Fil Rouge : Transformation automatisée de la donnée en connaissance

Simon ADDA - Mastère SIO - 2021/2022



CentraleSupélec

Table des matières

Introduction : 3

Objectifs : 3

Le dossier projetfilrouge : 3

Architecture : 5

1. Module Extraction 5

2. Module Extraction 6

3. Module ONTO 6

Modèles de NLP : 6

NLTK : 8

SPACY : 8

TEXTBLOB : 9

Résultats : 9

Relations entre auteurs et références : 10

Les services : 10

AWS : 10

SOA : 10

Docker : 10

Hadoop : 11

Conclusion : 11

Introduction :

Objectifs :

Les objectifs pour ce projet fil rouge sont les suivants :

« Étudier et développer l'ensemble d'une chaîne de traitements en Python, de la collecte des données en passant par la validation, la reconnaissance d'entités nommées, la mise en relation, la restitution, la déduction de nouvelles données, l'interrogation, la représentation de la connaissance produite. Être en mesure d'apporter un indicateur de la qualité et de la précision de la chaîne de traitement fait partie des objectifs. »

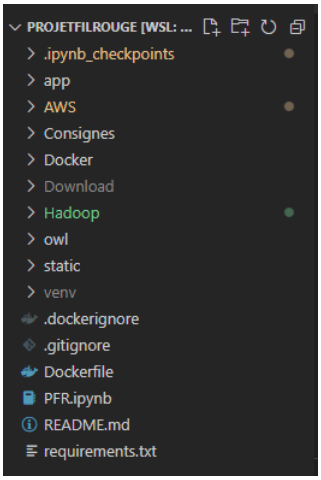
Il est donc essentiel, comme dans tous projets, de bien décomposer chaque brique de ce projet afin de bien répondre aux objectifs. On peut alors découper notre projet en 3 parties :

1. **Module Extraction des PDF :** Extraction du texte brut des PDF du site
2. **Module Intelligence artificielle :** Reconnaissance et extraction des noms de personnes et de relations entre elles
3. **Module Ontologie :** Constitution d'un graphe de connaissances

Afin de répondre au mieux aux enjeux de projet, l'utilisation du langage python a été essentiel avec notamment l'utilisation des packages FastAPI, Spacy ou encore Owlready mais également les technologies comme Docker ou les API.

Le dossier projetfilrouge :

Dans ce dossier vous trouverez les différents éléments permettant de répondre à l'ensemble du projet.



Dans chaque répertoire dédié à un service, vous trouverez un Readme.md permettant de vous guider dans l'exécution des services :

- Cours AWS : /AWS :
  Ici vous trouverez toutes les informations permettant de répondre à la problématique du cours AWS, sois l'utilisation d'un service AWS (cf. readme.md).
- Cours Hadoop : /Hadoop
  Dans ce répertoire, les scripts permanent d'importer une table sous format csv dans un cluster puis l'utilisation de Zeppelin afin d'analyser ces données.
- Cours Docker : /Docker
  Nous allons ici utiliser Docker pour l'utilisation des micro services via les conteneurs. L'application se trouvant dans /app est lors dans un conteneur et il est possible de l'exécuter directement (cf. readme.md)
- Cours SOA : /app
  Documenter la solution en utilisant le formalisme C4 abordé en cours. Il contient également une documentation avec le standard OpenAPI de l'API du projet (cf. readme.md)
- Ontologie dans le notebook PFR.ipynb
- Requirements.txt : Toutes les dépendances nécessaires au bon fonctionnement du projet (pour les services Docker, AWS et SOA)

Le projet est accessible également sous https://github.com/SimonADDA/PFR

Ce rapport explique en détail la partie Ontologie et explique brièvement les différents autres services mis en place.

Architecture :

1. Module Extraction

L'objectif en sortie de ce module est d'extraire les textes issus des PDF de toutes les publications en « Computer Science & AI » du site ArXiv.org.

Pour répondre à cette problématique, il est toujours important d'étudier l'existant notamment les différents packages python permettant de nous aider dans notre objectif. Apres avoir utilisé le package **BeautifuSoup** et l'API de Arvix.org (https://arxiv.org/help/api), j'ai décidé de m'orienter vers le package **Arvix** (https://pypi.org/project/arxiv/) qui permet d'extraire les liens, les auteurs et les titres de PDF issues de notre recherche.

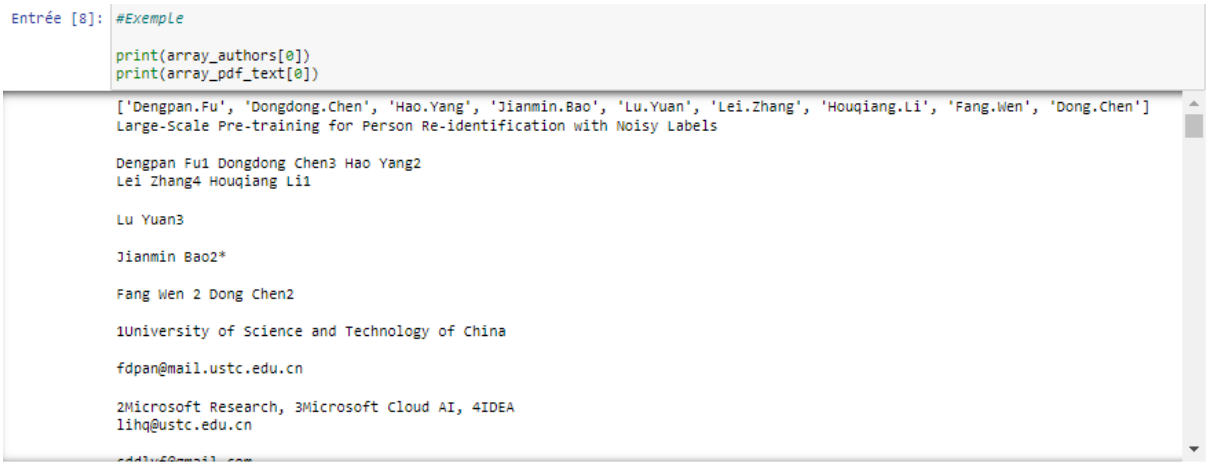
Entrée [4]:
```
#Store in array all informations on pdf (title, authors and links)

array_link=[]
array_authors=[]
array_title=[]

for result in search.results():
    array_link.append(result.pdf_url)
    array_title.append(result.title)
    temp=result.authors
    array_authors.append([re.sub("[^A-Za-z0-9]",".",str(i)) for i in temp])

    #Download pdf as link.pdf
    result.download_pdf(dirpath=".\Download", filename=f'{result.pdf_url[21:]}.pdf')

print("Link: ",array_link)
print("Authors: ",array_authors)
print("Title: ",array_title)
```

```
Link:  ['http://arxiv.org/pdf/2203.16533v1', 'http://arxiv.org/pdf/2203.16531v1', 'http://arxiv.org/pdf/2203.16530v1', 'http://
arxiv.org/pdf/2203.16529v1', 'http://arxiv.org/pdf/2203.16528v1']
Authors:  [['Dengpan.Fu', 'Dongdong.Chen', 'Hao.Yang', 'Jianmin.Bao', 'Lu.Yuan', 'Lei.Zhang', 'Houqiang.Li', 'Fang.Wen', 'Dong.
Chen'], ['Shengyi.Qian', 'Linyi.Jin', 'Chris.Rockwell', 'Siyi.Chen', 'David.F..Fouhey'], ['Yuliang.Zou', 'Zizhao.Zhang', 'Chun.
Liang.Li', 'Han.Zhang', 'Tomas.Pfister', 'Jia.Bin.Huang'], ['Jiahui.Lei', 'Kostas.Daniilidis'], ['Osman.Erman.Okman', 'Mehmet.G
orkem.Ulkar', 'Gulnur.Selda.Uyanik']]
Title:  ['Large-Scale Pre-training for Person Re-identification with Noisy Labels', 'Understanding 3D Object Articulation in In
ternet Videos', 'Learning Instance-Specific Adaptation for Cross-Domain Segmentation', 'CaDeX: Learning Canonical Deformation C
oordinate Space for Dynamic Surface Representation via Neural Homeomorphism', 'L^3U-net: Low-Latency Lightweight U-net Based Im
age Segmentation Model for Parallel CNN Processors']
```

Exemple d'utilisation du package Arvix pour cinq PDF

Les différents PDF seront alors téléchargées en local dans le répertoire */Download* puis exploitées grâce au package **Pdfminer** (https://pypi.org/project/pdfminer.six/) et sa fonction extract_text(). Dans un premier temps, j'ai utilisé le package **Pdftotext** mais ce dernier demande une installation un peu longue sous Windows.

Nous avons donc en sortie de ce premier module, les textes et les auteurs de nos PDF téléchargés ce qui nous sera très utiles pour la suite.



Exemple de texte issue d'un pdf accompagné de ses auteurs

2. Module Extraction

Dans ce second module, nous allons nous focaliser sur la partie extractions des entités nommées. Cette partie du projet était la plus complexe car elle demandait un vrai travail de comparaison entre les nombreux algorithmes existants. De plus, il est assez compliqué de comparer ces algorithme manières cartésienne et concrète. L'utilisation du service Compehend de AWS a alors permis d'avoir une bonne base de comparaison en partant du postulat que ce dernier fournissait un pourcentage de réussite d'extraction supérieur à celui des algorithmes déjà existant et libre d'accès comme NLTK, Spacy ou Textblob. (cf. Service AWS)

L'utilisation de ces algorithmes a donc amenés a des résultats parfois concluant et parfois décevant.

3. Module ONTO

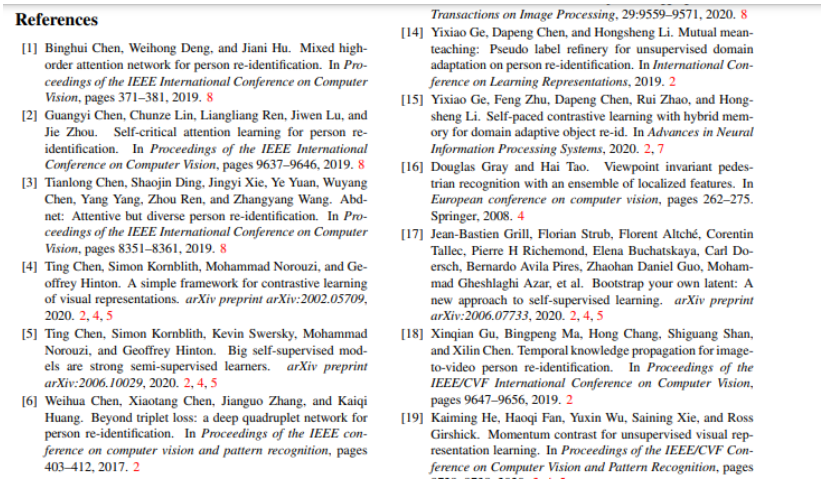
Dans ce module, nous allons mettre en place des relations entre les auteurs des articles et les personnes citées dans ces articles. L'utilisation du module python owlready va nous permettre de créer ces relations directement sur Python. Il est ensuite conseillé d'utiliser le logiciel Protégé afin d'étudier le graph de connaissances mis en place. Une fois le programme **PFR.ipynb** complétement implémenté, un fichier**. owl** sera accessible dans le répertoire **/owl.**

Modèles de NLP :

Avant d'implémenter les différents modèles de NLP, il est important de faire un pré traitement de nos textes issus de nos PDF afin d'exploiter au mieux ces algorithmes. L'idée était alors d'améliorer la performance de notre extraction d'entités nommées en effectuant un pre-process :

1. **Extraction du texte à partir du mot « References » :**

   L'idée ici est d'exploiter une partie du texte. En effet nous avons pour objectifs d'extraire les références d'un texte. Il est donc utile de s'intéresser à la partie du PDF ou les références sont tous réunis.



References

[1] Binghui Chen, Weihong Deng, and Jiani Hu. Mixed high-order attention network for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 371–381, 2019. 8

[2] Guangyi Chen, Chunze Lin, Liangliang Ren, Jiwen Lu, and Jie Zhou. Self-critical attention learning for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9637–9646, 2019. 8

[3] Tianlong Chen, Shaojin Ding, Jingyi Xie, Ye Yuan, Wuyang Chen, Yang Yang, Zhou Ren, and Zhangyang Wang. Abd-net: Attentive but diverse person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8351–8361, 2019. 8

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 2, 4, 5

[5] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020. 2, 4, 5

[6] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 403–412, 2017. 2

Transactions on Image Processing, 29:9559–9571, 2020. 8

[14] Yixiao Ge, Dapeng Chen, and Hongsheng Li. Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification. In *International Conference on Learning Representations*, 2019. 2

[15] Yixiao Ge, Feng Zhu, Dapeng Chen, Rui Zhao, and Hongsheng Li. Self-paced contrastive learning with hybrid memory for domain adaptive object re-id. In *Advances in Neural Information Processing Systems*, 2020. 2, 7

[16] Douglas Gray and Hai Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *European conference on computer vision*, pages 262–275. Springer, 2008. 4

[17] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 2, 4, 5

[18] Xinqian Gu, Bingpeng Ma, Hong Chang, Shiguang Shan, and Xilin Chen. Temporal knowledge propagation for image-to-video person re-identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9647–9656, 2019. 2

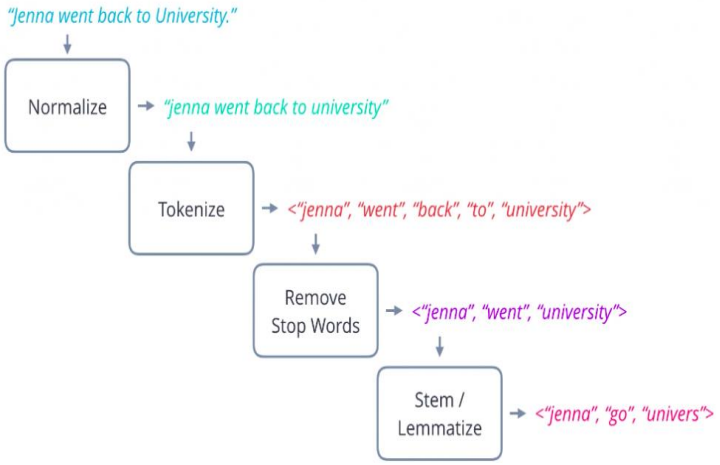
[19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages

Exemple de PDF et ses références

2. **Nettoyage du texte :**

   Une fois les références récupérées, il est nécessaire d'ajouter un pré traitement sur ce texte. L'utilisation d'une fonction preprocess_text () va donc nous être utile car elle réunira tous les filtres nécessaires.

   - Suppression des nombres
   - Suppression des lettres seules
   - Suppression des espaces
   - Utilisation de la tokenisation, lemmatisation et suppressions de stops words :



Utilisation de la tokenisation, des stop words et de la lemmatisation

Une fois ces différents nettoyages effectués, nous allons utiliser nos différents algorithmes afin d'en extraire les entités nommées.

NLTK :

NLTK est une plate-forme leader pour la création de programmes Python pour travailler avec des données de langage humain. Il fournit des interfaces faciles à utiliser à plus de 50 corpus et ressources lexicales telles que WordNet, ainsi qu'une suite de bibliothèques de traitement de texte pour la classification, la tokenisation, la radicalisation, le balisage, l'analyse et le raisonnement sémantique, des wrappers pour les bibliothèques NLP de puissance industrielle, et un forum de discussion actif (https://www.nltk.org/).

Dans un premier temps, après avoir étudier en détail ce package, j'ai décidé de passer par la fonction **get_human_names()** qui permet d'extraire les noms d'un texte en utilisant notamment une précision remarquable.

Dans un second temps, j'ai mis en place une API permettant d'extraire les entités nommées avec l'utilisation des packages additionnels :

- nltk("maxent_ne_chunker")
- nltk('punkt')
- nltk('averaged_perceptron_tagger')

Ces derniers permettent simplement d'utiliser différents fonctions fournis sur NLTK afin d'améliorer la performance de sortie de nos algorithmes.

SPACY :

spaCy est conçu pour aider à faire un travail d'analyse de texte, pour créer de vrais produits ou recueillir de vraies informations. (https://spacy.io/)

Le second algorithme implémenté est donc celui de Spacy avec notamment l'utilisation de **spacy.load('en_core_web_sm')** qui permet une utilisation rapide de notre extraction d'entités nommées. L'utilisation de **spacy.load("en_core_web_trf")** aurait pu être également utile mais le temps de traitement pose un problème sur le pipeline de l'ensemble du projet.

Deux fonctions ont alors été implémentées pour répondre à notre besoin. La première fonction nlp_entities(text), s'appuie simplement sur une récupération des mots ayant pour classe grammatical PROPN. La seconde fonction, extract(text) s'appuie quant à elle sur les token ["ART", "EVE", "NAT", "PERSON"].

Cette seconde fonction a permis d'apporter une efficacité supérieur a celle de la première fonction grâce notamment à sa précision sur les classes grammaticales et sur son pré traitement en mettant les mots en lowercase.

```
def extract(text:str) :
    spacy_nlp = spacy.load('en_core_web_sm')
    doc = spacy_nlp(text.strip())
    named_entities = []

    for i in doc.ents:
        entry = str(i.lemma_).lower()
        text = text.replace(str(i).lower(), "")
        if i.label_ in ["ART", "EVE", "NAT", "PERSON"]:
            named_entities.append(entry.title().replace(" ", "_").replace("\n","_"))
        named_entities = list(dict.fromkeys(named_entities))
    return named_entities
```

```
Entrée [21]: #Without preprocess and With preprocess

print("Len with preprocess",len(extract(references_clean)), "and without",len(extract(references)))
print("Ex:",extract(references_clean)[0])
```

```
Len with preprocess 187 and without 226
Ex: Binghui_Chen
```

```
Entrée [22]: #Result
extract(references_clean)
```

```
Out[22]: ['Binghui_Chen',
 'Weihong_Deng_Jiani_Hu',
 'Mixed',
 'Guangyi_Chen_Chunze',
 'Lin_Liangliang',
 'Jiwen_Lu',
 'Jie_Zhou',
 'Tianlong_Chen',
 'Ding_Jingyi_Xie_Ye',
 'Yuan_Wuyang',
 'Chen_Yang_Yang',
 'Zhou_Ren_Zhangyang_Wang',
 'Abd_In_Pro',
 'Ting_Chen',
 'Simon_Kornblith_Mohammad_Norouzi_Ge',
 'Ting_Chen_Simon_Kornblith_Kevin_Swersky_Mohammad_Norouzi_Geoffrey_Hinton',
 'Weihua_Chen_Xiaotang',
```

Exemple de sortie d'entités nommées d'un PDF avec Spacy

TEXTBLOB :

Le troisième et dernier algorithme utilisé est Texblob. Ce dernier est une bibliothèque Python pour le traitement de données textuelles. Il fournit une API simple pour plonger dans les tâches courantes de traitement du langage naturel (NLP) telles que le balisage des parties du discours, l'extraction de phrases nominales, l'analyse des sentiments, la classification, la traduction, etc. (https://textblob.readthedocs.io/en/dev/)

Pour notre projet, l'utilisation de cet algorithme fait simplement partie de ceux qui sont les moins efficaces en ayant en sortie de mots qui ne sont pas des noms le plus souvent du temps et apporte donc des fausses informations à notre graph de connaissances.

Résultats :

Bien évidemment, des tests ont été effectuées avec ces différents packages. Pour ce faire, ces tests ont été fait sur une dizaine de PDF en comptant manuellement le nombres d'entités nommées et en analysant les résultats fournis par les algorithmes. Il a donc été décidé de se positionner sur l'algorithme de Spacy et la seconde fonction implémenter : **extract().** Nous avons en sortie de cette algorithme des résultats plus que concluant avec, pour les PDF utilisée en exemple, une grande partie d'entre eux retrouvé dans nos extractions faites manuellement.

Relations entre auteurs et références :

Dans cette dernière partie de ce projet fil rouge, l'objectif est de mettre en place des relations entre les auteurs des PDF et les références de ces derniers afin d'aliment notre graph de connaissance.

```
Entrée [28]: from owlready import *

             onto_path.append("owl")

Entrée [29]: onto = Ontology("http://test.org/onto.owl")

             * Owlready * Creating new ontology onto <http://test.org/onto.owl>.

Entrée [30]: class Author(Thing):
                 ontology = onto

             class References(Thing):
                 ontology = onto

             class quoted_by(Property):
                 ontolgy = onto
                 domain = [References]
                 range = [Author]
```

Création des classes Authors, References et de la relation quoted_by

Une fois les auteurs et les références stockés dans des listes python, nous pouvons maintenant exploiter le package owlready pour concevoir notre graph de connaissance. Ce dernier est visible dans Protégé.

Les services :

/AWS :

Dans le répertoire **/AWS** vous trouverez des scripts python à implémenter permettant l'extraction des entités nommées de PDF avec l'utilisation du service comprehend. Les entités nommées sont alors fournies dans un fichier json dans les dossier **/Download** ou **/Professeur**.

On y trouve deux scripts python et un notebook (que l'utilisateur peut exécuter cellules par cellules afin de voir les résultats en détails AWS_service.ipynb).

Implémentation : faire un git clone https://github.com/SimonADDA/PFR.git puis suivre le **Readme.md** dans ce répertoire fourni les instruction d'implémentation.

/SOA :

Dans le répertoire **/app** vous trouverez une application sous forme d'API permettant de récupérer les tags d'un texte et d'extraire les entités nommées d'un texte en utilisant NLTK et FastAPI. On y trouve également le fichier api.json qui fournit le standard OpenAPI pour cette API. Des explications détaillées sont expliqué dans le pdf : **Service SOA.pdf**

Implémentation : faire un git clone https://github.com/SimonADDA/PFR.git puis suivre le **Readme.md** dans ce répertoire fourni les instruction d'implémentation.

/Docker :

Ce service permet de créer un conteneur l'application expliqué précédemment avec le fichier **Dockerfile**.

Implémentation : faire un git clone https://github.com/SimonADDA/PFR.git puis suivre le **Readme.md** dans ce répertoire fourni les instruction d'implémentation.

/Hadoop :

Dans ce répertoire se trouve deux scripts python ainsi qu'un fichier .csv. Le premier script **write_csv.py**, permet d'extraire les metadata des PDF et de les stocker dans la fichier **dataset.csv.** Le second script python, **csv_to_hdfs.py** va permettre d'envoyer ce fichier .csv directement dans un cluster sous HDFS afin d'en exploiter les données avec Zeppelin.

Implémentation : faire un git clone https://github.com/SimonADDA/PFR.git puis suivre le **Readme.md** dans ce répertoire fourni les instruction d'implémentation.

Conclusion :

Ce Projet Fil Rouge était particulièrement intéressant surtout pour moi qui avait assez peu utilisé les différents modèles de NLP et les graphs de connaissances durant mes études. Je souhaite de plus m'orienter vers ce genre de problématiques dans le futur et notamment durant mon stage. Il a fallu, pour réussir ce projet, bien comprendre et décomposer les différents objectifs et savoir comment implémenter ces différentes solutions afin d'avoir un pipeline fonctionnel.