Centrale Supélec

Rapport Projet Fil Rouge

MS SIO 2022

Jérémy Cao
08/04/2022

Table des matières

Introduction

Ce document contiendra mes réflexions et mes observations sur mes différences choix techniques.

Le projet est disponible sur Github à cette adresse :

https://github.com/Caojerem/Projet_fil_rouge_SIO_2022

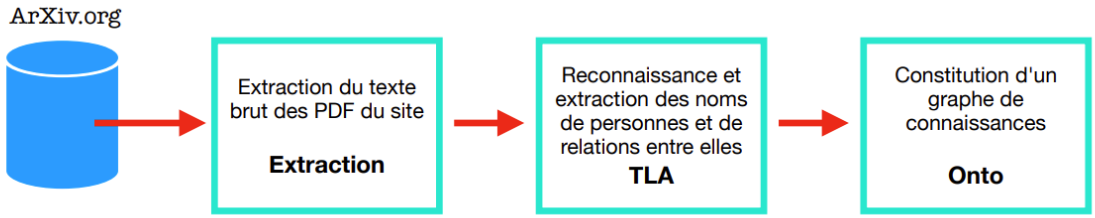
Les procédures d'installation sont indiquées dans le README.

Description du Notebook

Pour le Notebook, j'ai décidé d'aller au plus simple pour que l'exécution soit la plus courte possible sans imposer les données d'entrées ou de sorties. Le notebook récupère donc un seul PDF, l'un des plus récents. Les métadonnées sont récupérées (Titre, Auteur, Lien url). Le texte est extrait puis analysé. Les références sont parcourues jusqu'à trouver une référence provenant de Arxiv afin de la récupérer également pour avoir directement un lien pertinent pour l'ontologie. Les entités nommées sont ensuite traitées sur les deux PDF récupérés. L'ontologie est ensuite créée avec les auteurs et les références que l'on a limité à 2 (2 auteurs et 2 références par PDF) pour avoir un retour visuel propre.

Architecture

L'architecture se décompose tel que l'énoncé l'a décrit : un module de récupération des fichier PDF sur Arxiv, un module d'extraction de données et de traitement des entités nommées et un module d'ontologie.



Cette chaine de traitement est réalisée sur un Notebook.

Module de récupération des PDF sur Arxiv

J'ai utilisé une librairie appelée « Arxiv » qui m'a permis de me soustraire à l'exercice de « web scraping » et qui me retournait directement l'url ou les urls des PDF pertinents de la catégorie « Computer science & AI ».

Utilisation de la librairie ArXiv pou récupérer le lien des pdf

In [3]:
```
search = arxiv.Search(
    query = "computer science &  ai",
    # id_list=["1605.08386v1"]
    max_results = 2,
    sort_by = arxiv.SortCriterion.SubmittedDate
)

for result in search.results():
    print ("Titre :", result.title)
    print ("Auteurs :", result.authors)
    print ("Lien :", result.pdf_url)
```

```
Titre : Temporal Alignment Networks for Long-term Video
Auteurs : [arxiv.Result.Author('Tengda Han'), arxiv.Result.Author('Weidi Xi
e'), arxiv.Result.Author('Andrew Zisserman')]
Lien : http://arxiv.org/pdf/2204.02968v1
Titre : Enhanced Direct Speech-to-Speech Translation Using Self-supervised Pr
e-training and Data Augmentation
Auteurs : [arxiv.Result.Author('Sravya Popuri'), arxiv.Result.Author('Peng-Je
n Chen'), arxiv.Result.Author('Changhan Wang'), arxiv.Result.Author('Juan Pin
o'), arxiv.Result.Author('Yossi Adi'), arxiv.Result.Author('Jiatao Gu'), arxi
v.Result.Author('Wei-Ning Hsu'), arxiv.Result.Author('Ann Lee')]
Lien : http://arxiv.org/pdf/2204.02967v1
```

J'ai ensuite utilisé le module Python « urllib » pour télécharger les PDF données.

Module d'extraction de données et traitement des entités nommées

Extraction de données

Pour l'extraction de données, j'ai utilisé un module Python nommé « pdfx » qui utilise le module Python « pdfminer ». J'ai décidé d'utiliser ce module car les modules « pdftotext » et « pdfminer » me donnaient des résultats qui ne me convenaient pas pour certains PDF, notamment pour les articles en

plusieurs colonnes. J'ai donc utilisé le module « pdfx » qui traitait correctement ce genre de documents. Les résultats étaient aussi globalement plus propres, ce qui m'a permis de me soustraire d'un traitement post-extraction.

Traitement des entités nommées

Pour le traitement des entités nommées, j'ai utilisé le module « spacy » car il me donnait les meilleurs résultats. Le module « nltk » me donnait moins de résultats mais qui étaient également moins pertinents. Le module « spacy » me ressortait plus de noms de personnes mais me donnait également les titres des articles.

Pour juger de la pertinence des résultats, j'ai utilisé deux méthodes :

- Une vérification manuelle en comparant les résultats directement sur les PDF
- Avec le service AWS Comprehend sur les premières références (limitation AWS)

Le service AWS étant très efficace, le résultats de la comparaison a été similaire à ma comparaison manuelle. Le module « spacy » semble ne pas repérer un nom sur 5. Je n'ai pas observé de corrélation entre les noms oubliés, sur la nationalité ou la longueur par exemple.

Module de construction d'une ontologie

Le module d'ontologie commence par un traitement de références récupérées afin de retirer les caractères non reconnus. L'ontologie est composée une seule classe « Person » et d'une seule propriété de classe « get_ideas_from ». Les auteurs sont des « Person » qui ont la propriété « get_ideas_from » liés aux références qui sont également des « Person ».

Une règles est créée :

Création de règles complexes pour l'ontologie

In [21]:
```
with onto:
    imp = Imp()
    imp.set_as_rule("Person(?x), Person(?y), Person(?z), \
                    DifferentFrom(?x, ?y), DifferentFrom(?y, ?z), DifferentFrom(?x, ?z),  \
                    gets_ideas_from(?x, ?y), gets_ideas_from(?y, ?z) -> gets_ideas_from(?x, ?z)")
```

Cette règle stipule que si une personne prend des idées d'une autre personne et que cette autre personne prend des idées d'une troisième personne. La première personne prend des idées de la troisième personne.

Après s'être assuré que les « Person » sont différents et avoir activé le raisonneur Pellet, le résultat suivant s'est affiché :

- Avant le raisonneur :

```
print(len(globals()[author_list[0]].gets_ideas_from))
globals()[author_list[0]].gets_ideas_from
```

```
4
[onto.R._J._Weiss, onto.C._Zhang, onto.Ann_Lee, onto.Peng_Jen_Chen]
```

- Après le raisonneur :

```
lues --ignore-imports /tmp/tmp1tdz1car
* Owlready * Adding relation onto.Jordi_Adell gets_ideas_from onto.Ann_Lee
* Owlready * Adding relation onto.Sravya_Popuri gets_ideas_from onto.Pd_Aguero
```

```
print(len(globals()[author_list[0]].gets_ideas_from))
globals()[author_list[0]].gets_ideas_from
```

```
5
[onto.R._J._Weiss,
 onto.C._Zhang,
 onto.Ann_Lee,
 onto.Peng_Jen_Chen,
 onto.Pd_Aguero]
```