TP: Programmer avec le Web Sémantique

Jean-Claude Moissinac
jean-claude.moissinac@telecom-paristech.fr

27 juin 2018

Dans ce TP, nous allons aborder la manipulation de connaissances exprimées en RDF. Nous commencerons par nous familiariser interactivement avec l'accès aux données de la bibliothèque nationale de France. Ensuite, nous aborderons l'accès par programme Python à ces données et différentes possibilités d'exploiter les données obtenues.

Installations préliminaires

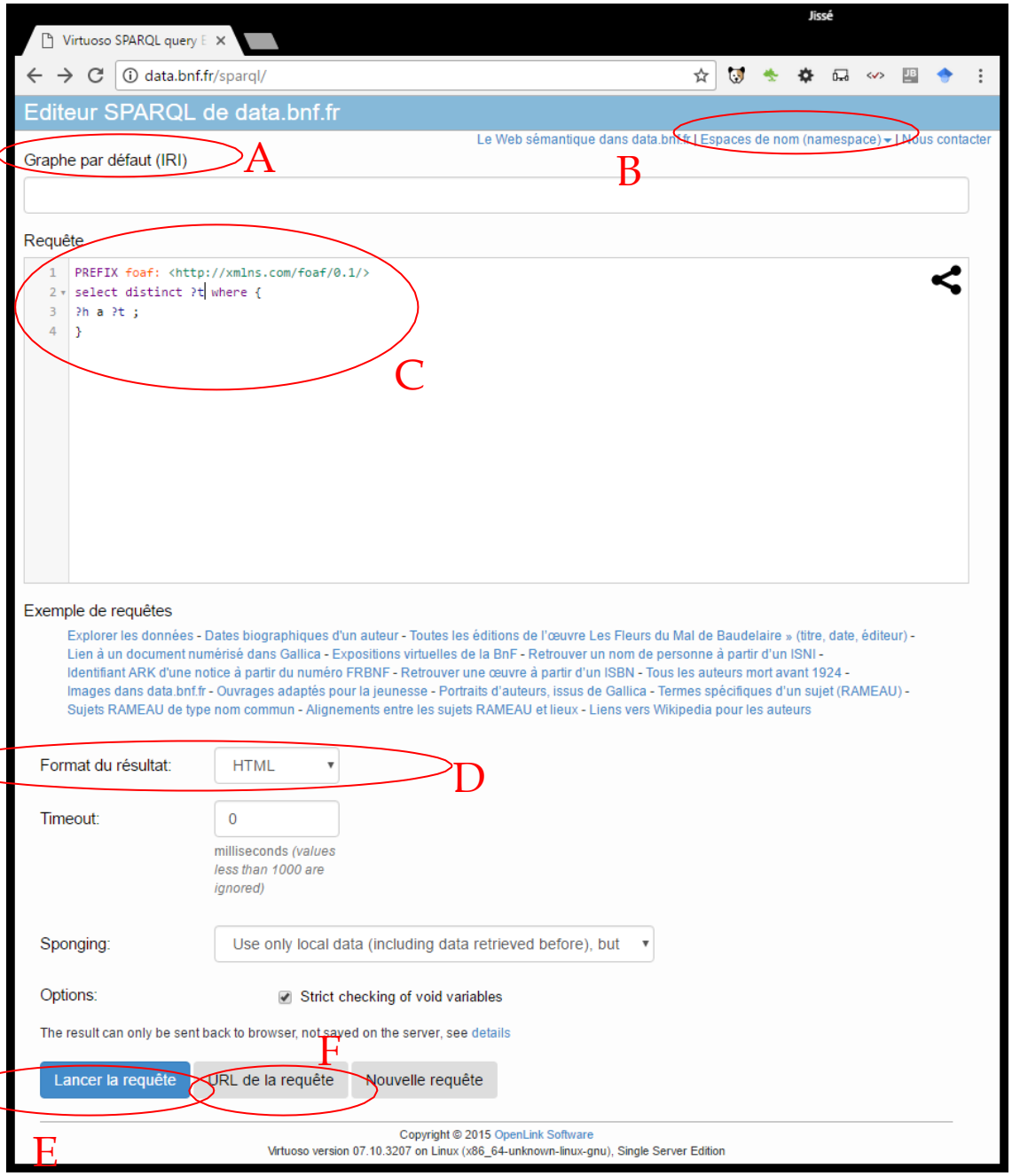
Le TP peut être réalisé sur tout ordinateur comportant une installation de Python 3.x, Firefox ou tout autre navigateur, ainsi que le paquets Python rdflib (installables via les commandes pip install ou easy_install). Il est vivement recommandé de créer un environnement virtuel avant de faire l'installation des outils:

```
python3 –m venv ./nomDossierPourVotreEnvironnement
source nomDossierPourVotreEnvironnement/bib/activate
```

1 Découverte interactive du point d'accès SPARQL de la BNF

Le point d'accès SPARQL de la BNF est à l'adresse :
http://data.bnf.fr/sparql/
Il s'agit à la fois d'une page de formulaire qui vous permet de tester des requêtes SPARQL et aussi du point d'accès qui permet d'envoyer des requêtes depuis un programme.
Dans la capture d'écran ci-après, les zones suivantes sont mises en évidence :
A La zone 'Graphe par défaut' permet d'indiquer un graphe de connaissances qui servira par défaut lorsqu'aucun graphe n'est indiqué dans la requête
B Ce lien permet d'afficher une liste de préfixes prédéfinis pour vous, afin de rendre les requêtes plus compactes et lisibles
C Zone d'édition des requêtes SPARQL proprement dite
D Cette liste donne les formats possibles pour le résultat des requêtes
E Exécution de la requête
F Devrait permettre la récupération de l'URL associée à la requête, pour obtenir le résultat sans passer par le présent formulaire ; on récupèrera cette URL dans la barre de

navigation du navigateur lorsque le résultat de la requête est affiché. On voit en bas de la fenêtre que le graphe de connaissances de la BNF est rendu accessible avec le logiciel Virtuoso.



1. Trouver les types de données présents dans data.bnf

La requête
select distinct ?t where { ?h a ?t }
permet de trouver tous les triplets qui relient une entité à un type (le prédicat 'a' est équivalent à 'rdf :type') et de sélectionner comme résultat tous les types distincts trouvés.(rem : certaines entités peuvent ne pas avoir de type)
La requête
select ?t (count(?h) as ?c) where { ?h a ?t } Group
by ?t order by ?c
permet de compter les occurrences de ces types.
Quels sont les cinq types qui apparaissent les plus représentatifs du contenu de la base?
Tentez d'expliquer la construction de la requête.
Par curiosité, vous pouvez cliquer sur les URI obtenues pour voir comment elles sont décrites.

2. Trouver des personnes, les compter, trouver les propriétés qui les décrivent

Nous allons nous intéresser aux entités de type foaf :Person.
Cherchez la valeur du préfixe foaf : et donnez la version développée du type foaf:Person. Nous allons compter les personnes décrites dans data.bnf. Pour cela, construisez une requête en vous inspirant de la question 1.
Pour les prochaines requêtes, pour alléger la lecture du sujet, nous omettrons les déclarations de préfixes ; vous pourrez, par exemple, chercher les préfixes sur le site prefix.cc.
Nous allons maintenant chercher quelques personnes décrites :
select distinct ?h where { ?h a foaf:Person } LIMIT 10 OFFSET 100
Vous pouvez cliquer sur les URI obtenues pour le résultat en format HTML : qu'est-ce que cela donne ? Pourquoi ? Notez l'url apparaissant dans la barre de navigation du navigateur et comparez-la à celle sur laquelle vous avez cliqué.
Nous allons chercher les propriétés –prédicats- qui sont utilisées pour décrire des personnes dans data.bnf

```
select distinct ?p where {
  ?h a foaf:Person;
    ?p [ ]
}
```

Notez la ligne
?p [ ]
Qui pourrait être remplacée par
?p ?o
La notation [ ] permet d'indiquer au moteur d'exécution de la requête qu'on ne

s'intéresse pas à la valeur associée à ?p, on s'intéresse juste à ?p.
Combien de prédicats sont utilisés ? Comment avez-vous obtenu ce résultat ?

Cliquez sur un des prédicats trouvés, par exemple
http://data.bnf.fr/ontology/bnf-onto/mortPourLaFrance
Au vu de ce que vous obtenez, expliquez pourquoi il est intéressant de rendre une URI déréférençable ? (c'est-à-dire que d'une façon ou d'une autre, elle envoie sur une page web explicative de ce que l'URI représente, ce qui n'est pas obligatoire)

Nous allons maintenant utiliser l'URI de la première personne obtenues au début de cette question et chercher comment cette personne est décrite dans data.bnf

La requête
describe ?h where { ?h a foaf:Person } LIMIT 1 OFFSET 100
vous donne une description ; pouvez-vous interpréter cette description : combien de 'faits' sont disponibles au sujet de cette personne ? quelles sont les deux grandes parties de la description ? quelle est l'année de naissance la personne?

Regardez ce que la requête donne avec un résultat en JSON. Commentez ce que vous pourriez faire de ce résultat en Python. Cliquez sur 'Url de la requête' et copiez cette URL.

Regardez ce que donne la requête au format NTriples. Pouvez-vous interpréter ce que vous voyez ?
Même démarche au format CSV ou TSV. Qu'y a-t-il de particulier ?

3. Répétez la démarche pour des documents

Avec la même démarche, vous pouvez découvrir la représentation dans data.bnf des entités de type foaf :Document
(et si vous êtes curieux, vous pouvez continuer avec d'autres types découverts à l'étape 1)

Voyons s'il y a des relations directes entre des documents et des personnes
```
select distinct ?p where {
  ?doc a foaf:Document ;
    ?p ?pers .
  ?pers a foaf:Person
}
```
Commentez cette requête. Quelle est la nature de la ou des relations trouvées ?

Comment interprétez-vous cette relation ?
Par exemple, comment cette relation est-elle utilisée avec la personne dont l'URI est http://data.bnf.fr/ark:/12148/cb11896834h#about? Trouvez le(s) document(s) avec cette relation vers cette personne. Interprétez le lien. Peut-on obtenir d'autres éléments descriptifs de ce document ?

4. Interroger data.bnf en Python

Vous allez maintenant soit seulement avec l'objet request, soit en installant l'objet sparqlWrapper effectuer des appels à data.bnf.
Vous serez amené à utiliser aussi l'objet json de python.
Commencez par tenter d'appeler l'URL sauvée à l'étape 2.
Quel résultat obtenez-vous dans la réponse à la requête ?
Modifiez la requête pour définir le format souhaité pour la réponse. Vous pouvez utiliser la librairie urllib et notamment la méthode urllib.parse.urlencode pour mettre les paramètres de la requête dans un format acceptable pour une URL.
Vous pouvez créer un modèle de requête avec une partie variable en vous servant de la méthode replace des chaines de caractère.
Ex
modele = « select * where { ?s ?p _name }"
query1 = modele.replace("_name", "Paris")
query2 = modele.replace("_name", "Lyon")
Obtenez la réponse en JSON et exploitez le résultat dans votre code Python, par exemple en imprimant des traces ou en générant un fichier html qui contiendra le résultat mis en forme dans une page web (si vous vous sentez à l'aise en HTML).
Essayez diverses requêtes utilisées précédemment ou des variantes de celles-ci que vous générerez grâce à votre programme Python.

5. Au-delà de la BNF (bonus)

Sur le principe de l'utilisation de données 'sémantiques' depuis un langage de programmation, imaginez une utilisation que vous pourriez faire d'un des ensembles de données suivants :

- Dbpedia http://dbpedia.org/sparql
- IMDB http://data.linkedmdb.org
- HAL https://data.archives-ouvertes.fr/doc/sparql
- Ou d'un jeu de données que vous choisirez ici : https://lod-cloud.net/
  Ou là : https://www.w3.org/wiki/SparqlEndpoints