Examen - web sémantique

Mercredi 16 novembre 2020

(durée : 1h00 – barême : 20 points. Les TP seront noté chacun sur 10. La note de l'examen aura un coefficient de 0,7 et celle des TP de 0,3)

1. **Questions de cours** (répondre de manière synthétique.. 4 phrases maximum par réponse) **(5pts)**

1.1 Les graphes de connaissance RDF et RDFS (2 pts)

1.1.1 Comment la ressource rdfs:Class permet-elle de distinguer les différents types de ressources d'un graphe RDF ?

Par la classe rdfs :Class, on peut distinguer certaines ressources et les déclarer comme des classes : Ci rdf :type rdfs :Class. Cela veut dire que ces ressources Ci représentent des ensembles de ressources r représentant des entités, et qui sont du type de ces classes : r rdf :type Ci

1.1.2 Quelle est la différence entre rdf:type et rdfs:subClassOf ?

rdf :type associe un type à une ressource (la ressource appartient à l'ensemble du type) ; rdfs :subClassOf relie 2 entités de type rdfs :Class, l'une étant plus spécifique que l'autre (relation d'inclusion entre 2 ensembles)

1.2 Utilisation des données liées ouvertes (3pts)

1.2.1 Quel est l'intérêt d'utiliser l'URI représentant Paris de DBPedia (dbr :Paris) dans une application plutôt que la chaîne de caractère « Paris » pour indiquer une adresse par exemple ?

Grâce à la ressource dbr:Paris on a moins d'ambiguïté qu'avec la chaîne de caractère Paris (qui pourrait être un nom commun ou une ville d'un autre pays de la France). Un autre avantage est de pouvoir collecter des propriétés de Paris, des connaissances grâce aux properties associées à dbr :Paris dans DBPedia.

1.2.2 Expliquez 2 manières dont un moteur de recherche peut utiliser les données liées ou les bases de connaissances disponibles sous forme de données liées.

1. Pour désambiguiser les requêtes en LN, mieux « comprendre » sur quoi porte la requête, le moteur peut chercher la ou les meilleures ressources dans un ensemble de données liées qui peut être reconnue à partir des mots de la requête.
2. En réponse à la requête, pour restituer de l'information synthétique au sujet d'une ou des entités reconnues ; 3. pour faciliter la recherche cross-lingue.

2. – Exercice 2 (10 pts, 2 par question) : FUNCTIONAL et contraintes de cardinalité

Dans l'ontologie du cinéma sur laquelle vous avez travaillé en TP1, on trouve les classes Personne, Acteur et Genre, associées aux data type et object Properties suivants :

```
tp1:Personne a owl :Class ;
        rdfs:subClassOf [ rdf:type owl:Restriction ;
                          owl:onProperty :aGenre ;
                          owl:cardinality "1"^^xsd:nonNegativeInteger
                        ] ;
```

```
tp1:Masculin a tp1:genre .
tp1:Féminin a tp1:genre .
tp1:Genre a owl :Class .
tp1:Film a owl:Class .
tp1:Artiste a tp1:Personne .

tp1:aGenre a owl:ObjectProperty ;
       rdfs:range tp1:Genre ;
       rdfs:domain tp1:Personne .
tp1:aRéalisé a owl:ObjectProperty ;
        rdfs:range tp1:Film ;
        rdfs:domain tp1:Personne ;
        owl:inverseOf :réaliséPar .
tp1:joueDansFilm a owl:ObjectProperty ;
           rdfs:range :Film ;
           rdfs:domain :Personne .

tp1:Homme a owl:Class ;
      owl:equivalentClass [ rdf:type owl:Class ;
                    owl:intersectionOf ( tp1:Personne
                                  [ a owl:Restriction ;
                                    owl:onProperty tp1:aGenre ;
                                    owl:hasValue tp1:Masculin
                                  ]
                                 )
                   ] .

tp1:Acteur a owl:Class ;
       a tp1:Artiste ;
       rdfs:label "Actor"@en , "Acteur"@fr , "Comédien"@fr ;
       owl:equivalentClass [a  owl:Class ;
                     owl:intersectionOf ( tp1:Homme
                                   [ a owl:Restriction ;
                                     owl:onProperty tp1:joueDansFilm ;
                                     owl:onClass tp1:Film ;
                                     owl:minQualifiedCardinality "1"^^xsd:nonNegativeInteger
                                   ]
                                  )
                    ] .
tp1:Realisateur a owl:Class ;
          owl:equivalentClass [ a owl:Class ;
                        owl:intersectionOf ( tp1:Personne
                                [ a owl:Restriction ;
                                  owl:onProperty tp1:aRéalisé ;
                                  owl:onClass tp1:Film ;
                                  owl:minQualifiedCardinality 1"^^xsd:nonNegativeInteger
                                ]
                               )
                       ] .
```

2.1. Définir, en utilisant la syntaxte Turtle , deux entités tp1:Woody_Allen et tp1:Manhattan de type owl:Class ; représenter que tp1:Woody_Allen joue dans Manhattan et a réalisé Manhattan.

```
tp1:Woody_Allen     a owl:Class ;
                    tp1:aRéalisé tp1:Manhattan ;
                    tp1:joueDansFilm tp1:Manhattan .
tp1:Manhattan a owl:Class .
```

2.2. tp1:Woody_Allen sera-t-il classé comme tp1:Acteur ? pourquoi ?

non : il sera classé tp1:Personne car c'est le domaine de tp1:joueDansFilm et de tp1:aRéalisé.

Il ne sera pas classé comme tp1:Acteur car pour être reconnue comme étant de ce type, une entité doit être de type tp1:Homme. Et pour être un tp1:Homme, il faut posséder la property tp1:aGenre tp1:Masculin. Or on ne connait pas le genre de Woody Allen dans tp1. Il n'est donc ni un tp1:Homme ni un tp1:Acteur.

2.3. Si on ajoute à la base de connaissances tp1 deux sous-classes de tp1:Film : « court métrage » (film de durée inférieure à 60 mn) et « long métrage » (film de durée supérieure à 60 mn), on observe que le raisonneur classe tp1:Manhattan comme un tp1:CourtMetrage alors qu'on ne lui a pas donné de durée. Quelle hypothèse et quelles règles conduisent le raisonneur à affecter cette classe plutôt que tp1:Film ?

Comme la durée de tp1:Manhattan n'est pas renseignée, le système la considère sans doute comme null et donc <60. De plus, tp1:Manhattan est classé comme un film à cause du triplet tp1:Woody_Allen tp1:aRéalisé tp1:Manhattan par exemple. Il est donc conforme à la définition de tp1:courtMétrage.

Règles du raisonneur : inférer la classe la plus précise possible.

FUNCTIONAL : le cours comporte une erreur sur sa définition. **Functional indique qu'une propriété relie une ressource à AU PLUS une autre ressource** (et non exactement 1).

2.4. Quelle est la différence entre le fait d'indiquer qu'une tp1:Personne a un unique lien vers une tp1:genre par la propriété tp1:aGenre comme dans l'encadré ci-dessus ET le fait de définir tp1:aGenre comme Functional ?

Si on indique que la cardinalité de tp1:aGenre est 1 exactement, alors toute personne doit avoir un et un seul genre (au moins et au plus un). Si la propriété est Functional, toute entité reliée par tp1:aGenre à un genre peut avoir 0 ou 1 genre.

2.5. Si on associe 2 genres (tp1:Masculin et tp1:Feminin) à l'entité tp1:Woody_Allen, le raisonneur ne constate pas d'erreur à cause de l'hypothèse du monde ouvert. Il considère à partir de là que les deux entités tp1:Masculin et tp1:Feminin sont identiques. Quelles nouvelles classes seront inférées pour tp1:Woody_Allen ? quelle relation poser entre tp1:Masculin et tp1:Feminin pour éviter cela ?

tp1:Woody_Allen sera classé come tp1:Homme, tp1:Acteur grâce au genre tp1:Masculin, en plus de tp1:Personne et tp1:Réalisateur. Pour l'éviter, il faut déclarer que les 2 entités tp1:Masculin et tp1:Feminin sont diffentes avec la property owl:differentFrom (réponse tolérée : owl:disjointWith)

3. Exercice 3 (5pts) : Utiliser COUNT et GROUP BY dans SPARQL

Dans la base de connaissances de la BNF, les trois éléments suivants contribuent à décrire des ouvrages présentant l'œuvre de la Jean de la Fontaine « la cigale et la fourmi »

```
<http://data.bnf.fr/11975188/jean_de_la_fontaine_la_cigale_et_la_fourmi/studies>
dcterms:subject <http://data.bnf.fr/ark:/12148/cb119751881#frbr:Work> .

<http://data.bnf.fr/ark:/12148/cb119751881> a skos:Concept ;
   bnf-onto:FRBNF 11975188 ;
   dcterms:created "1985-07-06" ;
   dcterms:modified "2016-01-27" ;
   rdfs:seeAlso <http://catalogue.bnf.fr/ark:/12148/cb119751881> ;
   skos:note "Fable appartenant au premier recueil, publié sous le titre \"Fables choisies mises
en vers par monsieur de La Fontaine\""@fr ;
   skos:prefLabel "La cigale et la fourmi"@fr ;
   foaf:focus <http://data.bnf.fr/ark:/12148/cb119751881#frbr:Work> .

<http://data.bnf.fr/ark:/12148/cb119751881#frbr:Work> a frbr:Work ;
   rdfs:label "La cigale et la fourmi" ;
   bnf-onto:firstYear 1668 ;
   bnf-onto:subject "Littératures" ;
   dcterms:creator <http://data.bnf.fr/ark:/12148/cb11910267w#foaf:Person> ;
   dcterms:date "1668" ;
   dcterms:description "Fable appartenant au premier recueil, publié sous le titre \"Fables
choisies mises en vers par monsieur de La Fontaine\""@fr ;
   dcterms:language <http://id.loc.gov/vocabulary/iso639-2/fre> ;
   dcterms:subject <http://dewey.info/class/800/> ;
   dcterms:title "La cigale et la fourmi"@fr ;
   rdagroup1elements:dateOfWork <http://data.bnf.fr/date/1668/> ;
   ore:isAggregatedBy <http://data.bnf.fr/ark:/12148/cb120083695#frbr:Work> ;
   = <http://data.bnf.fr/ark:/12148/cb119751881#about> ;
   foaf:depiction <http://gallica.bnf.fr/ark:/12148/bpt6k129255c.thumbnail>,
      <http://gallica.bnf.fr/ark:/12148/bpt6k54338578.thumbnail>,
      <http://gallica.bnf.fr/ark:/12148/bpt6k58388000.thumbnail>,

<https://upload.wikimedia.org/wikipedia/commons/6/6d/The_Ant_and_the_Grasshopper_by_
Charles_H._Bennett.jpg> .
```

3.1 **Interprétation de ces définitions.** Le premier bloc définit un skos:concept qui renvoie à une notice de la BNF (une entrée de leur catalogue), et à laquelle on peut associer une œuvre qui va se matérialiser dans plusieurs ouvrages (qui ont chacun un éditeur et un aspect physique différent, mais parfois même une écriture différente, en vieux français ou français plus contemporain par exemple) (2 pts)

3.1.1 Quel est le type qui permet de représenter une œuvre dans le vocabulaire frbr ? Quelle est l'objectProperty qui lie une entrée de catalogue et une œuvre ?

frbr :Work

foaf :focus (pas dcterm:subjet car relie la notice à l'oeuvre ; pas rdfs:seeAlso car relie l'entrée catalogue à une autre entité de catalogue.bnf.gov qui n'est pas la même entité que l'euvre)

3.1.2 Sachant que la propriété dcterms:creator lie une œuvre à son auteur, quel est le concept qui, dans cette base de connaissances, représenterait Jean de la Fontaine ?
<http://data.bnf.fr/ark:/12148/cb11910267w#foaf:Person>

3.2 **Requêtes SPARQL sur ces données**. Rappel de la syntaxe de COUNT et GROUP BY pour compter le nombre d'entités en lien avec une autre entité : la requête suivante affiche, pour chaque film, le nombre d'acteurs connus dans la base tp1 comme jouant dans ce film. (3 pts)

```
SELECT ?film (COUNT (distinct ?acteur) as ?count)
    WHERE { ?acteur tp1:joueDansFilm ?film }
GROUP BY ?acteur
```

3.2.1 Ecrire une requête qui afficherait tous les titres des œuvres de Jean de la Fontaine dans data.bnf.fr et leur langue à côté.

```
Prefix tp1 : …
Prefix rdfs : …
Prefix dcterms : ..
SELECT ?titre ? lang
        WHERE { ?oeuvre a frbr:Work .
        ?oeuvre dcterms:creator http://data.bnf.fr/ark:/12148/cb11910267w#foaf:Person .
        ?oeuvre dcterms:language ?lang .
        ?oeuvre dcterms :title ?titre . }
SORT BY ?titre
```

3.2.2 Ecrire une requête qui compte toutes les œuvres de tous les auteurs de la base (elle affiche, pour chaque auteur, le nombre de ces œuvres).

```
Prefix tp1 : …
Prefix rdfs : …
Prefix dcterms : ..
SELECT ?nomauteur ?auteur(COUNT (distinct ?oeuvre) as ?count)
        WHERE {
        ?oeuvre a frbr:Work .
        ?oeuvre dcterm:creator ?auteur .
OPTIONAL (?auteur foaf:name ?nomauteur . ) }
GROUP BY ?oeuvre
```