Vers une approche pour la reformulation automatique de requêtes à partir d'alignements complexes

Élodie Thiéblin, Fabien Amarger, Ollivier Haemmerlé, Nathalie Hernandez, Cassia Trojahn

IRIT, Institut de Recheche en Informatique de Toulouse, Toulouse, France
elodie.thieblin@gmail.com,{fabien.amarger, ollivier.haemmerle, nathalie.hernandez, cassia.trojahn}@irit.fr

Résumé : Cet article présente une approche de reformulation automatique de requêtes SPARQL à partir d'un alignement complexe entre deux ontologies. Poser une requête initialement écrite pour une base de connaissances donnée sur une nouvelle base implique de la reformuler à partir du vocabulaire employé dans cette dernière. Un alignement simple permet d'identifier des correspondances entre un élément de la première ontologie et un élément de la deuxième. Ce type d'alignement n'est souvent pas adapté pour exprimer l'ensemble des correspondances pouvant être établies entre deux ontologies. Un alignement complexe permet en effet de déterminer des correspondances plus fines entre 1 ou n éléments de la première ontologie et 1 ou n éléments de la deuxième. Peu d'approches considèrent ce type d'alignements alors qu'il paraît indispensable pour croiser la connaissance provenant des différentes bases de connaissances dont le vocabulaire repose à l'échelle du web de données liés sur des modélisations différentes. Notre approche repose sur des règles de transformation permettant de traduire automatiquement des requêtes SPARQL de type SELECT à partir d'alignements complexes exprimés en EDOAL. Aucun jeu de données ne permettant à notre connaissance de tester ce type d'approche, nous proposons également dans ce papier deux jeux de données que nous avons constitués. Les règles de réécriture ont été validées sur ces jeux de données.
Mots-clés : alignement d'ontologies, alignements complexes, SPARQL, réécriture de requêtes

1 Introduction

Les sources présentes sur le web de données liées sont très hétéroclites. Quand bien même leur format est identique et conforme aux exigences du W3C (RDF, RDFS et OWL), elles restent très hétérogènes par leurs modèles et vocabulaires. Le langage SPARQL est le langage d'interrogation de données RDF. Une requête en SPARQL est propre à la source qu'elle se destine à interroger. Pour cela, la requête SPARQL dépend du modèle ontologique à la base de la source RDF. Si un utilisateur veut poser la même question sur une autre base de connaissances (ou source), il doit adapter la requête SPARQL à cette source.

Combler l'écart d'hétérogénéité terminologique et sémantique entre bases de connaissances devient indispensable pour profiter pleinement du potentiel du web de données liées. L'alignement d'ontologies (Euzenat & Shvaiko, 2007) est une solution à cet enjeu grandissant. Il existe plusieurs types d'alignements : les alignements simples et les alignements complexes. Les alignements simples font correspondre un à un les éléments ontologiques équivalents sémantiquement dans les deux ontologies. Cependant, cette méthode d'alignement ne peut couvrir tous les cas d'utilisation à cause des différences de modèles entre sources ontologiques. Les alignements complexes pallient les faiblesses des alignements simples. Ils étendent les correspondances simples à des correspondances entre constructions complexes d'entités ontologiques.

Les alignements simples sont facilement exploitables lors de la traduction de requêtes

SPARQL. La démarche générale, intégrée à l'Alignment API (David *et al.*, 2011), par exemple, consiste à remplacer l'IRI d'une entité de la requête initiale par l'IRI qui lui correspond dans l'alignement simple, en considérant que la correspondance établie désigne une relation d'équivalence. Cependant, ne considérer que ce type de correspondances ne permet de traduire qu'un ensemble limité de requêtes SPARQL. Dans cet article, nous proposons une approche visant à exploiter les alignements complexes lors de la reformulation de requêtes SPARQL. L'objectif est de proposer un mécanisme s'adaptant au mieux à l'interrogation de sources hétérogènes. Nous souhaitons favoriser le croisement de connaissances et l'utilisation du Web de données liées. Même si peu de systèmes permettent aujourd'hui de générer automatiquement des alignements complexes, établir manuellement ce type de correspondances utilisées en entrée de notre approche est une tâche qui peut s'avérer moins fastidieuse que de reformuler manuellement chacune des requêtes SPARQL potentielles pour un nouvel entrepôt.

Notre approche propose des règles de reformulation pour les requêtes SPARQL de type SELECT à partir des correspondances complexes, impliquant une relation d'équivalence, et exprimables dans la syntaxe EDOAL. Nous proposons également deux jeux de données sur lesquels notre approche a été validée. Le premier correspond à un jeu de données construit pour répondre aux besoins d'experts en agriculture souhaitant trouver sur DBpedia des connaissances complémentaires à celles décrites dans une base de connaissances portant sur les taxons agronomiques. Le second a été conçu en vue d'enrichir le jeu de données de tâche OA4Q de OAEI [1] et porte sur un sous-ensemble du jeu de données en lien avec l'organisation de conférences.

L'article est structuré comme suit. Dans un premier temps, nous définissons les alignements, correspondances et leur formalisation (§2). Ensuite, nous présentons un état de l'art de la reformulation de requêtes SPARQL (§3). Les règles de transformation sur lesquelles repose notre approche (§4) ainsi que les jeux de données sur lesquels elle a été validée sont également présentés (§5).

2 Alignement d'ontologies

Un **alignement** A entre une ontologie source $\mathcal{O}$ et une ontologie cible $\mathcal{O}'$ est un ensemble de correspondances $\{c_1, c_2, ..., c_n\}$ entre $\mathcal{O}$ et $\mathcal{O}'$. A est directionnel et se note $A_{\mathcal{O}\rightarrow\mathcal{O}'}$ (Gillet *et al.*, 2013). Une correspondance c_i est un triplet $\langle e_{\mathcal{O}}, e_{\mathcal{O}'}, r\rangle$:

— si la correspondance est simple, $e_{\mathcal{O}}$ fait référence à une et une seule entité (classe ou propriété) de $\mathcal{O}$ et $e_{\mathcal{O}}$ à une et une seule entité de $\mathcal{O}'$. La correspondance est donc de cardinalité (1:1) ;

— sinon la correspondance est complexe. $e_{\mathcal{O}}$ représente un sous-ensemble d'éléments $\in \mathcal{O}$ et $e_{\mathcal{O}'}$ un sous-ensemble d'éléments $\in \mathcal{O}'$. Les éléments de $e_{\mathcal{O}}$ (resp. $e_{\mathcal{O}'}$) sont reliés entre eux en utilisant un langage formel tel que la Logique du Premier Ordre ou les Logiques de Description. La correspondance est alors de cardinalité (1:n,m:1,m:n).

— r est une relation entre ces deux entités $e_{\mathcal{O}}$ et $e_{\mathcal{O}'}$ telle que l'équivalence ($\equiv$), plus spécifique ($\sqsubseteq$) ou plus général ($\sqsupseteq$).

Une correspondance $\langle e_{\mathcal{O}}, e_{\mathcal{O}'}, r\rangle$ est unique dans un alignement $A_{\mathcal{O}\rightarrow\mathcal{O}'}$. Un alignement est dit complexe si au moins une de ses correspondances est complexe.

1. `http://oaei.ontologymatching.org/2015/conference/index.html`

La syntaxe EDOAL (Euzenat *et al.*, 2007; David *et al.*, 2011) permet d'exprimer les correspondances simples et complexes de cardinalité (1:1), (1:n), (n:1) et (n:m) en précisant la relation entre deux entités ($\sqsupseteq$, $\sqsubseteq$ ou $\equiv$). Les entités e_i et e'_i d'une correspondance complexe c_i sont représentées par des expressions (expressions de classe CE, de relation RE ou de propriété PE) qui peuvent être un ID (ou URI), une construction (composition de plusieurs entités liées par des opérateurs) ou une restriction. Pour une description détaillée des constructeurs définis dans la syntaxe EDOAL, nous renvoyons le lecteur vers (Euzenat *et al.*, 2007).

Nous illustrons par la suite cette syntaxe à partir de correspondances complexes établies entre les ontologies ekaw [2] pour laquelle nous utilisons le préfixe ekaw : <"http://ekaw#">, cmt [3] pour laquelle nous utilisons le préfixe cmt : <"http://cmt#"> et confOf [4] pour laquelle nous utilisons le préfixe confOf : <"http://confOf#">. Ces exemples présentent un fragment d'EDOAL impliquant des relations d'équivalence entre les entités. L'exemple 1 présente une **expression de classe** exprimant que la classe $Chairman$ de l'ontologie cmt est équivalente à l'union des classes $Demo\_Chair$, OC_Chair, PC_Chair, $Session\_Chair$, $Tutorial\_Chair$ et $Workshop\_Chair$ de l'ontologie ekaw. L'exemple 2 établit que la classe $Accepted\_Paper$ de ekaw est équivalente à **l'expression de classe** composée d'éléments ontologiques de cmt restreignant la classe $Paper$ aux individus pour lesquels le domaine de la relation $hasDecision$ est un individu de type $Acceptance$.

Exemple 1

```
<entity1>
  <edoal:Class rdf:about="&cmt;Chairman"/>
</entity1>
<entity2>
<edoal:Class>
  <edoal:or rdf:parseType="Collection">
    <edoal:Class rdf:about="&ekaw;Demo_Chair"/>
    <edoal:Class rdf:about="&ekaw;OC_Chair"/>
    <edoal:Class rdf:about="&ekaw;PC_Chair"/>
    <edoal:Class rdf:about="&ekaw;Session_Chair"/>
    <edoal:Class rdf:about="&ekaw;Tutorial_Chair"/>
    <edoal:Class rdf:about="&ekaw;Workshop_Chair"/>
  </edoal:or>
</entity2>
<measure rdf:datatype="&xsd;float">1.0</measure>
<relation>Equivalence</relation>
```

Exemple 2

```
<entity1>
    <edoal:Class rdf:about="&ekaw;Accepted_Paper"/>
</entity1>
<entity2>
  <edoal:Class>
    <edoal:and rdf:parseType="Collection">
      <edoal:Class rdf:about="&cmt;Paper"/>
        <edoal:AttributeDomainRestriction>
          <edoal:onAttribute>
            <edoal:Relation rdf:about="&cmt;hasDecision"/>
          </edoal:onAttribute>
          <edoal:class>
            <edoal:Class rdf:about="&cmt;Acceptance"/>
          </edoal:class>
        </edoal:AttributeDomainRestriction>
    </edoal:and>
  </edoal:Class>
</entity2>
<measure rdf:datatype="&xsd;float">1.0</measure>
<relation>Equivalence</relation>
```

Dans l'exemple 3, la relation $writtenBy$ de ekaw est équivalente à **l'expression de relation** définissant l'inverse de la relation $writePaper$. L'exemple 4 montre une correspondance entre la classe $Early - Registered\_Participant$ de l'ontologie ekaw et une **expression de classe** définie pour contraindre la classe $Participant$ de l'ontologie confOf à partir d'une **expression de propriété** restreignant la valeur de la propriété $earlyRegistration$ de ses instances à la valeur $true$.

2. `http://oaei.ontologymatching.org/2015/conference/data/ekaw.owl`
3. `http://oaei.ontologymatching.org/2015/conference/data/cmt.owl`
4. `http://oaei.ontologymatching.org/2015/conference/data/confOf.owl`

Exemple 3

```
<entity1>
  <edoal:Relation rdf:about="&ekaw;writtenBy"/>
</entity1>
<entity2>
  <edoal:Relation>
    <edoal:inverse>
      <edoal:Relation rdf:about="&cmt;writePaper"/>
    </edoal:inverse>
  </edoal:Relation>
</entity2>
<measure rdf:datatype="&xsd;float">1.0</measure>
<relation>Equivalence</relation>
```

Exemple 4

```
<entity1>
  <edoal:Class rdf:about="&ekaw;Early-Registered_Participant"/>
</entity1>
<entity2>
  <edoal:Class>
    <edoal:and rdf:parseType="Collection">
      <edoal:Class rdf:about="&confOf;Participant"/>
      <edoal:AttributeValueRestriction>
        <edoal:onAttribute>
          <edoal:Property rdf:about="&confOf;earlyRegistration"/>
        </edoal:onAttribute>
        <edoal:comparator rdf:resource="&edoal;equals"/>
        <edoal:value>
          <edoal:Literal edoal:type="xsd;boolean" edoal:string="true"/>
        </edoal:value>
      </edoal:AttributeValueRestriction>
    </edoal:and>
  </edoal:Class>
</entity2>
<measure rdf:datatype="&xsd;float">1.0</measure>
<relation>Equivalence</relation>
```

3 État de l'art

La réécriture de requêtes SPARQL se fait généralement à partir d'alignements simples. En termes généraux, cela consiste à remplacer l'IRI d'une entité de la requête initiale par l'IRI qui lui correspond dans l'alignement. Cette approche, intégrée à l'Alignment API (David *et al.*, 2011), ne prend pas en compte la relation de correspondance entre les deux entités (généralisation, spécialisation, équivalence). Dans (Euzenat *et al.*, 2008), la combinaison d'extensions de SPARQL pour la réécriture de requêtes SPARQL du type construct, en exploitant les alignements complexes a été proposée. La méthode est appliquée dans un contexte de transformation d'instances entre différents entrepôts de données. Une approche de réécriture ne se restreignant pas à construct et fondée sur des alignements plus expressifs a été proposée par Correndo *et al.* (Correndo *et al.*, 2010), travail dans lequel un ensemble de règles de réécriture de sous-graphes RDF a été défini. Cette approche emploie une formalisation déclarative des alignements entre deux structures RDF. Dans (Correndo & Shadbolt, 2011), un sous-ensemble d'expressions EDOAL sont traduites en leurs patrons de réécriture. Les expressions impliquant les restrictions sur les concepts ou relations, les propriétés différentes de l'égalité et les restrictions sur les occurrences de propriétés ne sont pas prises en compte. Zheng *et al.* (Zheng *et al.*, 2012) présentent une approche de réécriture en se fondant sur la notion de contexte correspondant aux différentes hypothèses sur la façon dont les alignements peuvent être interprétés. Un algorithme de réécriture de triplets RDF prend en compte ces différents contextes et résout de potentiels conflits entre eux. Makris *et al.* (Makris *et al.*, 2010, 2012) présentent le système de réécriture SPARQL-RW qui prend en compte un sous-ensemble de types de correspondances complexes fondés sur une représentation en Logique de Description (i.e., *ClassExpression*, *ObjectPropertyExpression*, *Datatype Property*, et *Individual*). L'algorithme est fondé sur la réécriture de patrons de graphes RDF qui consiste à substituer à un patron initial un patron final en conservant toute variable présente dans le graphe initial. Finalement, Gillet *et al.* (Gillet *et al.*, 2013) proposent une approche de réécriture de patrons de requêtes qui caractérisent des familles de requêtes. L'approche prend en compte des alignements simples et complexes mais n'exploite

pas l'expressivité d'EDOAL.

Dans cet article, nous proposons un ensemble de règles pour la réécriture automatique de requêtes SPARQL, fondé sur des alignements complexes. Contrairement à (Correndo & Shadbolt, 2011), l'approche supporte les restrictions sur les concepts et relations. Cependant, l'approche est limitée aux alignements complexes 1:n et ne prend en compte ni les requêtes sources utilisant les filtres, ni les unions, ni les options SPARQL. Notre approche se fonde sur l'hypothèse selon laquelle les requêtes à reformuler ont pour but de trouver de nouvelles instances répondant à un besoin. Pour cette raison, seuls les éléments de la $\mathcal{T}box$ (informations terminologiques d'une base de connaissances) sont alignés : les classes, les propriétés sur les objets et les propriétés sur les données. La démarche présentée dans cet article cherche à mettre à profit les différentes possibilités d'expressions d'éléments offertes par la syntaxe EDOAL. Comparativement à l'approche de Makris (2010), EDOAL est une alternative permettant la représentation d'alignements plus expressifs. Comme pour les travaux ci-dessous, la génération d'alignements complexes dépasse le cadre de cet article. Très peu d'approches ont éte proposées (Dhamankar *et al.*, 2004; Ritze *et al.*, 2010; Meilicke *et al.*, 2013) et très peu de systèmes gérant les correspondances complexes sont disponibles ou utilisent EDOAL pour les représenter.

4 Approche de reformulation de requêtes SPARQL

Dans nos travaux, nous nous concentrons sur la reformulation de requêtes SPARQL de type SELECT de la forme suivante :

$$R_{\mathcal{O}} \; = \; SELECT \; DISTINCT? \; (\; Var + \; | \; '*' \;) \; WHERE \; \{ \; T_{R_{\mathcal{O}}} \; \}$$

où T_{R_O} correspond à l'ensemble des triplets ou motifs de graphes exprimés dans la requête initiale à partir de l'ontologie $\mathcal{O}$. Un triplet t de cet ensemble est composé d'un sujet s, d'un prédicat p et d'un objet o.

$$\forall t \in T_{R_{\mathcal{O}}}, \quad t = \langle s, p, o \rangle$$

Notre approche vise à produire l'ensemble $T_{R_{O'}}$ contenant les triplets exprimés en fonction de l'ontologie O' à partir de T_{R_O} et de l'alignement complexe $A_{\mathcal{O}\rightarrow\mathcal{O}'}$. Les correspondances complexes (1:n) établissant des relations d'équivalence sont exploitées pour constituer $T_{R_{O'}}$. Les 3-uplets ci-dessous représentent la nature des entités impliquées dans les correspondances à partir des constructeurs EDOAL (constructions et restrictions) (Euzenat *et al.*, 2007) :

$$\langle ClassID, ClassID | ClassConstruction | ClassRestriction, \equiv \rangle$$
$$\langle RelationID, RelationID | RelationConstruction | RelationRestriction, \equiv \rangle$$
$$\langle PropertyID, PropertyID | PropertyConstruction | PropertyRestriction, \equiv \rangle$$

Notre approche analyse la nature des triplets de $T_{R_{\mathcal{O}}}$ en fonction des entités le composant et des correspondances établies dans l'alignement $A_{\mathcal{O}\rightarrow\mathcal{O}'}$. Nous supposons que l'alignement a établi toutes les correspondances nécessaires pour les entités de $T_{R_{\mathcal{O}}}$. Nous définissons des règles qui pour tout triplet de $T_{R_{\mathcal{O}'}}$ génèrent 1 ou n triplets dans $T_{R_{\mathcal{O}'}}$. Deux types de triplets sont considérés : les triplets de types *Classe* et les triplets de types *Relation*. Si un triplet n'est pas identifié comme étant d'un de ces types, il n'est pas traduit mais ajouté dans l'ensemble $T_{R_{\mathcal{O}'}}$.

4.1 Triplets classe

Les triplets classe, notés $T_{R_\mathcal{O}}^{Classe}$, sont de la forme

$$\forall t \in T_{R_\mathcal{O}}^{Classe} t = \langle s, p, o_\mathcal{O} \rangle \quad \text{, où} \begin{cases} s \text{ est une variable} \\ p \text{ est rdf:type} \\ o_\mathcal{O} \text{ est une } \textit{ClassID} \\ \exists < o_\mathcal{O}, o_{\mathcal{O}'}, \equiv > \in A_{\mathcal{O} \to \mathcal{O}'} \end{cases}$$

Un triplet classe est identifié si $o_\mathcal{O}$ est une *ClassID* et si il existe une correspondance impliquant $o_\mathcal{O}$ et une expression de classe $o_{\mathcal{O}'}$. Lors de la transformation d'un triplet classe $t_\mathcal{O}$, le sujet $s_\mathcal{O}$ et le prédicat $p_\mathcal{O}$ restent inchangés. Seul $o_\mathcal{O}$ est traduit suivant l'élément $o_{\mathcal{O}'}$ qui lui correspond dans l'alignement. Les règles de traduction sont fonction de l'expression associée à $o_{\mathcal{O}'}$:

1. *ClassID*: si l'expression de classe $o_{\mathcal{O}'}$ est une *ClassID*, le triplet à rajouter à $T_{R_{\mathcal{O}'}}$ sera

$$\langle s, p, o_{\mathcal{O}'} \rangle$$

2. *ClassConstruction*: Les *ClassExpression* composant $o_{\mathcal{O}'}$ sont annotées comme suit : $o_{\mathcal{O}'}^1$, $o_{\mathcal{O}'}^2$, etc. La transformation du triplet classe dépend de l'opérateur de la relation construction.
   (a) AND : l'intersection mène à l'insertion dans $T_{R_{\mathcal{O}'}}$ de plusieurs triplets ayant le même sujet :
   $$\{\langle s, p, o_{\mathcal{O}'}^1 \rangle \cap \langle s, p, o_{\mathcal{O}'}^2 \rangle \cap ...\}$$
   (b) OR : l'union de plusieurs relations est représentée en SPARQL par une "UNION" entre triplets ayant le même sujet :
   $$\{\langle s, p, o_{\mathcal{O}'}^1 \rangle \cup \langle s, p, o_{\mathcal{O}'}^2 \rangle \cup ...\}$$
   Le tableau 1 donne un exemple de transformation de requêtes à partir de l'exemple 1 de correspondance.
   (c) NOT : consiste à trouver l'ensemble des triplets $\langle s, p, v \rangle$, où v est une variable intermédiaire, duquel on supprime les triplets $\langle s, p, o_{\mathcal{O}'}^1 \rangle$.
   $$\{\langle s, p, v \rangle \text{ MINUS } (\langle s, p, o_{\mathcal{O}'}^1 \rangle) \}$$

3. *ClassRestriction* : les restrictions sur les expressions de classe prennent en compte des expressions de relation ou de propriété, notées $relation(o_f)$ ou $propriete(o_f)$. La transformation du triplet classe dépend de l'opérateur de l'expression correspondante :
   (a) *AttributeTypeRestriction*: cette restriction s'applique à une expression de propriété $o_{\mathcal{O}'}$ consistant à restreindre le type de donnée de cette propriété à un certain $type$. Pour la transformation du triplet de classe t, on ajoute à $T_{R_{\mathcal{O}'}}$ un triplet ayant pour sujet s, pour prédicat l'expression de propriété $propriete(o_{\mathcal{O}'})$ et pour objet une variable intermédiaire v. La restriction se fait sur v à l'aide d'un filtre SPARQL (FILTER) et de la fonction datatype(v).
   $$\{\langle s_i, propriete(o_f), v \rangle \text{ FILTER (datatype}(v) = type \text{) } \}$$

Requête pour cmt	Requête traduite pour ekaw

SELECT ?z WHERE {
?z rdf:type cmt:Chairman.
}
	SELECT ?z WHERE{
	{ ?z rdf:type ekaw:Demo_Chair. }
	UNION { ?z rdf:type ekaw:OC_Chair. }
	UNION { ?z rdf:type ekaw:PC_Chair. }
	UNION { ?z rdf:type ekaw:Session_Chair. }
	UNION { ?z rdf:type ekaw:Tutorial_Chair. }
	UNION { ?z rdf:type ekaw:Workshop_Chair. }
	}

TABLE 1 – Traduction d'un triplet type classe à partir d'une correspondance entre une classe et un expression de classe construite à partir d'une union donnée dans l'exemple 1

Requête pour ekaw	Requête traduite pour cmt

SELECT ?z WHERE {
?z rdf:type ekaw:Accepted_Paper.
}
	SELECT ?z WHERE{
	?z rdf:type cmt:Paper.
	?z cmt:hasDecision ?var_temp.
	?var_temp rdf:type cmt:Acceptance. }

TABLE 2 – Traduction d'un triplet type classe à partir d'une correspondance entre une classe et un expression de classe construite à partir d'une expression de classe donnée dans l'exemple 2

(b) *AttributeDomainRestriction*: cette restriction restreint le codomaine d'une relation issue de l'expression de classe $o_{\mathcal{O}'}$ à une expression de classe $codomaine(o_{\mathcal{O}'})$ elle aussi exprimée dans $o_{\mathcal{O}'}$. Pour traduire t, on ajoute à $T_{R_{\mathcal{O}'}}$ deux triplets. Le premier exprime la relation $relation(o_{\mathcal{O}'})$ entre le sujet s et une variable intermédiaire v. Le deuxième triplet assure que la classe de v soit bien l'expression de classe $codomaine(o_{\mathcal{O}'})$.

$$\{\langle s, relation(o_{\mathcal{O}'}), v\rangle \cap \langle v, rdf : type, codomaine(o_{\mathcal{O}'})\rangle\}$$

Le tableau 2 donne un exemple de transformation de requêtes à partir de l'exemple 2 de correspondance.

(c) *AttributeValueRestriction*: cette restriction s'applique à une expression de relation. Pour limiter les valeurs de l'objet du triplet $relation(o_{\mathcal{O}'})$ on utilise un filtre SPARQL (FILTER) sur la comparaison entre la variable v et la $valeur$ renseignée. Dans l'implémentation actuelle, la valeur renseignée ne peut être qu'un littéral ou une instance. Le comparateur cp correspond à l'un des comparateurs de la syntaxe EDOAL : "=", "<" et ">".

$$\{\langle s, relation(o_{\mathcal{O}'}), v\rangle \text{ FILTER } (v\ cp\ valeur)\},\ cp \in \{=, <, >\}$$

(d) *AttributeOccurenceRestriction*: cette restriction cherche à imposer une contrainte sur le nombre d'occurrences d'une relation ou propriété. Pour compter le nombre d'occurrences en question, on imbrique une sélection en SPARQL (SELECT) du sujet s ainsi que le compte $compte_v$ d'une variable intermédiaire v. Le compte est calculé grâce à la fonction SPARQL COUNT. Ce SELECT imbriqué porte sur le triplet $\langle s, relation(o_{\mathcal{O}'}), v\rangle$. Le compte $compte_v$ est ensuite comparé à la valeur de la restric-

tion val_{rest} grâce au comparateur cp dans un filtre SPARQL (FILTER).

$$\begin{array}{l}\{\{\text{SELECT } s \text{ (COUNT}(v) \text{ AS } compte_v) \text{ WHERE}\\ \{\langle s, relation(O_{\mathcal{O}'}), X\rangle\}\\ \text{GROUP BY } s.\ \}\\ \text{FILTER } (compte_v\ cp\ val_{rest})\},\\ cp \in \{=, <, >\}\end{array}$$

4.2 Triplets relation

Les triplets relation, notés $T^{Relation}_{R_{\mathcal{O}}}$, sont de la forme

$$\forall t \in T^{Relation}_{R_{\mathcal{O}}} t = \langle s, p_{\mathcal{O}}, o\rangle \quad \text{, où } \begin{cases} s \text{ une variable} \\ p_{\mathcal{O}} = \text{ une RelationId ou PropertyId} \\ o \text{ une variable ou un littéral} \\ \exists < p_{\mathcal{O}}, p_{\mathcal{O}'}, \equiv > \in A_{\mathcal{O}\to\mathcal{O}'} \end{cases}$$

Dans les triplets relation, $p_{\mathcal{O}}$ est une *PropertyId* et $p_{\mathcal{O}'}$ une expression de relation ou propriété. Lors de la transformation d'un triplet, la nature de l'expression est considérée :

1. *RelationId* ou *PropertyId*: le triplet suivant est inséré dans $T_{R_{\mathcal{O}'}}$

$$\langle s, p_{\mathcal{O}'}, o_i\rangle$$

2. *RelationConstruction* ou *PropertyConstruction*: la transformation dépend de l'opérateur de la construction. Les opérateurs suivis d'une * ne sont valables que pour les RelationConstructions. Les RelationExpressions ou PropertyExpressions composant $p_{\mathcal{O}'}$ seront indicées comme suit: $p^1_{\mathcal{O}'}$, $p^2_{\mathcal{O}'}$,...
   (a) AND : cette construction peut avoir un ou plusieurs composants *RelationExpression*. Les triplets générés seront :
   $$\{\langle s, p^1_{\mathcal{O}'}, o\rangle \cap \langle s, p^2_{\mathcal{O}'}, o\rangle \cap ...\}$$
   (b) OR : cette construction peut avoir un ou plusieurs composants *RelationExpression*. Les triplets générés seront :
   $$\{\langle s, p^1_{\mathcal{O}'}, o\rangle \cup \langle s, p^2_{\mathcal{O}'}, o\rangle \cup ...\}$$
   (c) NOT : la négation d'un triplet relation correspond à l'ensemble des triplets $\langle s, v, o\rangle$, où v est une variable, duquel on retranche les triplets $\langle s, p^1_{\mathcal{O}'}, o\rangle$. Les triplets générés seront :
   $$\{\langle s, v, o\rangle \text{ MINUS } (\ \langle s, p^1_{\mathcal{O}'}, o\rangle\ )\ \}$$
   (d) COMPOSE : une composition de relations est une chaîne de relations. Des variables intermédiaires v_1, v_2, ... sont introduites pour compléter la chaîne de relations entre le sujet s et l'objet o. On considère qu'une imbrication de composition ou qu'une imbrication d'une négation au sein d'une composition serait un problème de modélisation de

l'alignement. Cette *RelationConstruction* peut avoir un ou plusieurs composants *RelationExpression*. Dans la transformation, p est traduit par un ensemble de triplets de la forme :

$$\{\langle s, p^1_{\mathcal{O}'}, v_1\rangle \cap \langle v_1, p^2_{\mathcal{O}'}, v_2\rangle \cap ... \cap \langle v_{n-1}, p^n_{\mathcal{O}'}, o\rangle\}$$

(e) INVERSE* : inverser une relation revient à intervertir son sujet et son objet. Cette construction ne peut avoir qu'un composant *RelationExpression*. Le triplet généré sera :

$$\langle o, p^1_{\mathcal{O}'}, s\rangle$$

Le tableau 3 donne un exemple de transformation de requêtes à partir de la correspondance donnée en exemple 3 ;

(f) REFLEXIVE* : une relation réflexive est une relation entre le sujet s et lui-même. Sa représentation SPARQL est donc un triplet (sujet, relation, sujet). Elle ne peut avoir qu'un composant *RelationExpression*. Le triplet généré sera :

$$\langle s, p^1_{\mathcal{O}'}, s\rangle$$

(g) SYMMETRIC* : la symétrie d'une relation est l'union entre une relation et son inverse. Cette construction ne peut avoir qu'un composant *RelationExpression*. Les triplets générés seront :

$$\{\langle s, p^1_{\mathcal{O}'}, o\rangle \cup \langle o, p^1_{\mathcal{O}'}, s\rangle\}$$

3. *RelationDomainRestriction* ou *PropertyDomainRestriction*: pour restreindre le domaine d'une relation à une expression de classe, est rajoutée à $T_{R_{\mathcal{O}'}}$:

$$\{\langle s, p^1_{\mathcal{O}'}, o\rangle \cap \langle s, rdf : type, domain(p^1_{\mathcal{O}'})\rangle\}$$

4. *RelationCoDomainRestriction*: pour restreindre le codomaine d'une relation à une expression de classe, est rajoutée à $T_{R_{\mathcal{O}'}}$:

$$\{\langle s, p^1_{\mathcal{O}'}, o\rangle \cap \langle o, rdf : type, domain(p^1_{\mathcal{O}'})\rangle\}$$

5. *PropertyTypeRestriction*: pour restreindre le type de donnée d'une propriété, on utilise un filtre SPARQL (FILTER) agrémenté de la fonction "datatype(objet)" pour assurer l'égalité entre le $type$ souhaité et le type de donnée de l'objet : datatype(o).

$$\{\langle s, p^1_{\mathcal{O}'}, o\rangle \text{ FILTER (datatype}(o) = type)\}$$

6. *PropertyValueRestriction*: la restriction d'une propriété sur la valeur de son objet peut être représentée par un filtre SPARQL (FILTER) sur la comparaison entre l'objet de la propriété et la valeur renseignée. Dans la version actuelle, la valeur renseignée ne peut être qu'un littéral. Le comparateur cp correspond à l'un des comparateurs de la syntaxe EDOAL : "=", "<" et ">". Le tableau 4 donne un exemple de transformation d'une requête à partir d'une correspondance impliquant une expression de classe, elle-même définie à partir d'une expression de propriété restreignant la valeur, donnée en exemple 4.

$$\{\langle s, p^1_{\mathcal{O}'}, o_i\rangle \text{ FILTER } (o\ cp\ valeur)\},\ cp \in \{=, <, >\}$$

Requête pour ekaw	Requête pour cmt

SELECT ?z WHERE {
?paper :writtenBy ?author.
}
	SELECT ?z WHERE{
	?author cmt:writePaper ?paper.
	}

TABLE 3 – Traduction d'un triplet relation à partir de la correspondance de l'exemple 3

Requête pour ekaw	Requête pour confOf

SELECT ?z WHERE{
?z rdf:type ekaw:Early-Registered_Participant.
}
	SELECT ?z WHERE{
	?z rdf:type confOf:Participant.
	?z confOf:earlyRegistration ?var_temp.
	FILTER(?var_temp="true"^^ xsd:boolean).
	}

TABLE 4 – Traduction d'un triplet type classe impliquant une expression de classe définie à partir d'une expression de relation donnée dans l'exemple 4

5 Validation

À notre connaissance, aucun jeu de données intégrant deux bases de connaissances, l'alignement complexe entre les deux ontologies concernées et des requêtes SPARQL écrites pour les deux bases, n'est disponible. Seuls des jeux de données composés d'alignements simples existent dans le cadre de la tâche oa4qa [5] de la campagne OAEI. En reprenant le principe de cette tâche, nous avons constitué manuellement deux nouveaux jeux de données afin de valider notre approche.

Bases de connaissances et requêtes SPARQL. Le premier jeu de données a été construit lors d'un projet cherchant à accéder à des connaissances en lien avec une taxonomie des plantes. Pour répondre à ce besoin, les bases de connaissances Agronomic Taxon [6] et DBpedia ont été considérées. Il était plus précisément question de retrouver la connaissance suivante :

— $qa1$: les taxons de type espèce
— $qa2$: les taxons ayant pour rang taxinomique supérieur un taxon de type famille
— $qa3$: les taxons de rang taxinomique règne
— $qa4$: les taxons de rang taxinomique ordre
— $qa5$: les taxons de rang taxinomique genre

Pour constituer le jeu de données, les requêtes SPARQL de référence correspondant à ces besoins décrits en langage naturel ont été écrites manuellement pour chacune des deux bases de connaissances. L'objectif identique à celui suivi dans la tâche oa4qa est de pouvoir vérifier le fait qu'une requête reformulée automatiquement renvoie les mêmes résultats que la requête de référence.

Nous avons suivi la même démarche pour constituer le deuxième jeu de données. Il vise à interroger un sous-ensemble d'ontologies du jeu de données OAEI 2015 portant sur l'organisa-

5. http://oaei.ontologymatching.org/2015/
6. http://ontology.irstea.fr/AgronomicTaxon

Requête initiale posée sur ekaw	Requête de référence pour confOf	Requête générée pour confOf

SELECT ?person WHERE {
?person :authorOf ?paper.
?paper a :Paper.
?person rdf:type :Early-Registered_Participant.
}
	SELECT ?person WHERE{
	?person :earlyRegistration true.
	?person :writes ?papier.
	?papier a :Paper.
	}
		SELECT ?person WHERE {
		?person :writes ?paper.
		?paper a :Paper.
		?person rdf:type :Participant.
		?person :earlyRegistration ?variable_temp0.
		FILTER(?variable_temp0 = "true"^^xsd:boolean).
		}

TABLE 5 – Exemple de requête initiale, de requête de référence et de requête générée

tion de conférences [7]. Nous avons plus précisément considéré trois ontologies (cmt, confOf et ekaw). Nous avons défini en langage naturel les besoins suivants, à savoir retrouver :

— $qb1$: les reviewers de papiers acceptés
— $qb2$: les auteurs de soumissions longues
— $qb3$: les chairmen qui ont soumis un papier
— $qc1$: les participants qui se sont inscrits tôt et qui sont auteurs d'un papier soumis
— $qc2$: les participants qui se sont inscrits tard et qui ont écrit un poster

Les trois ontologies ont été peuplées avec des instances répondant à ces besoins. Parallèlement, les besoins ont été traduits en requêtes SPARQL écrites spécifiquement pour chacune des bases de connaissances considérées.

Les requêtes sur ces deux jeux de données sont disponibles en ligne [8].

Alignements complexes. Pour le jeu de données Agronomic Taxon et DBpedia, 10 correspondances complexes (et 1 simple) ont été produites manuellement. Pour le jeu de données Conférences, 8 correspondances simples et 6 correspondances complexes ont été construites manuellement. Les alignements sont disponibles en ligne [9].

Discussion. Le tableau 5 montre pour le besoin $qc1$, la requête SPARQL initialement posé sur $ekaw$, la requête de référence considérée pour la base $confOf$ et la requête générée par notre approche pour cette base à partir de l'exemple 4 et de la correspondance simple $ekaw : authorOf \equiv confOf : writes$. Bien que la requête de référence et la requête générée soient syntaxiquement différentes, leur exécution renvoie le même ensemble de résultats. Ceci est le cas pour l'ensemble des requêtes considérées. Les requêtes générées sont disponibles en ligne [10].

6 Conclusion et perspectives

Dans ce papier, nous avons présenté une approche qui exploite un alignement complexe entre deux ontologies pour réécrire des requêtes SPARQL formulées pour une ontologie source en

7. http://oaei.ontologymatching.org/2015/conference/index.html
8. https://www.irit.fr/recherches/MELODI/telechargements/requetes.zip
9. https://www.irit.fr/recherches/MELODI/telechargements/alignements.zip
10. https://www.irit.fr/recherches/MELODI/telechargements/requetesgenerees.zip

requêtes SPARQL formulées pour une ontologie cible. Une première validation de l'approche a permis de mettre en place deux jeux de données qui devront à très court terme être étoffés.

Les pistes d'amélioration sont nombreuses car le mécanisme de traduction se limite à des requêtes formatées, uniquement composées de triplets dont le sujet est une variable. Une autre piste de recherche serait la prise en compte des correspondances (n:m) d'un alignement complexe. Dans notre implémentation, la relation d'une correspondance entre deux entités (généralisation, spécialisation ou équivalence) n'est pas prise en compte. Une recherche sur leur signification lors d'une traduction de requête SPARQL peut être intéressante. Une autre extension possible consiste à considérer la transitivité des relations. Nous pouvons également améliorer certains points du mécanisme de traduction, notamment le fait que la *PropertyValueRestriction* est appliquée seulement sur des littéraux, ou que l'*AttributeValueRestriction* porte seulement sur les instances et les littéraux.

Références

CORRENDO G., SALVADORES M., MILLARD I., GLASER H. & SHADBOLT N. (2010). SPARQL Query Rewriting for Implementing Data Integration over Linked Data. In *1st International Workshop on Data Semantics (DataSem 2010)*.

CORRENDO G. & SHADBOLT N. (2011). Translating expressive ontology mappings into rewriting rules to implement query rewriting. In *6th Workshop on Ontology Matching*.

DAVID J., EUZENAT J., SCHARFFE F. & TROJAHN C. (2011). The Alignment API 4.0. *Semantic Web*, **2**(1), 3–10.

DHAMANKAR R., LEE Y., DOAN A., HALEVY A. & DOMINGOS P. (2004). imap: Discovering complex semantic matches between database schemas. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, p. 383–394.

EUZENAT J., POLLERES A. & SCHARFFE F. (2008). Processing ontology alignments with sparql. In *International Conference on Complex, Intelligent and Software Intensive Systems*, p. 913–917.

EUZENAT J., SCHARFFE F. & ZIMMERMANN A. (2007). *Expressive alignment language and implementation*. Rapport interne, INRIA.

EUZENAT J. & SHVAIKO P. (2007). *Ontology Matching*. Berlin, Heidelberg: Springer-Verlag.

GILLET P., TROJAHN C., HAEMMERLÉ O. & PRADEL C. (2013). Complex correspondences for query patterns rewriting. In *Proceedings of the 8th International Workshop on Ontology Matching*.

MAKRIS, GIODALSIS B. C. (2010). Ontology mapping and sparql rewriting for querying federated rdf data sources.

MAKRIS K., BIKAKIS N., GIOLDASIS N. & CHRISTODOULAKIS S. (2012). SPARQL-RW: transparent query access over mapped RDF data sources. In *15th International Conference on Extending Database Technology*, p. 610–613: ACM.

MAKRIS K., GIOLDASIS N., BIKAKIS N. & CHRISTODOULAKIS S. (2010). Ontology mapping and SPARQL rewriting for querying federated RDF data sources. In *2010 Conference on On the Move to Meaningful Internet Systems*, p. 1108–1117.

MEILICKE C., NOESSNER J. & STUCKENSCHMIDT H. (2013). Towards joint inference for complex ontology matching. In *Late-Breaking Developments in the Field of Artificial Intelligence*.

RITZE D., VÖLKER J., MEILICKE C. & SVÁB-ZAMAZAL O. (2010). Linguistic analysis for complex ontology matching. In *5th Workshop on Ontology Matching*.

ZHENG X., MADNICK S. E. & LI X. (2012). SPARQL Query Mediation over RDF Data Sources with Disparate Contexts. In *WWW Workshop on Linked Data on the Web*.