Génération de patrons de requêtes à partir d'alignements d'ontologies: Application à un système d'interrogation du Web sémantique fondé sur les patrons

Pascal GILLET

Master 2 Recherche
Informatique & Télécommunications
Recherche d'Information, Bases de Données & Multimédia

Institut de Recherche en Informatique de Toulouse
Université Paul Sabatier

Superviseurs: Cassia Trojahn, Ollivier Haemmerlé

Juin 2013

Résumé

Ce mémoire s'intéresse à l'utilisation d'alignements d'ontologies pour aider la tâche de réécriture automatique de patrons de requêtes. Cette approche s'inscrit dans le contexte du système SWIP, qui permet d'interroger des données RDF à partir de requêtes en langue naturelle, masquant ainsi la complexité du langage SPARQL. SWIP est fondé sur l'utilisation de patrons qui caractérisent des familles de requêtes, et qui sont instanciés en fonction de l'ensemble des mots-clés dans les requêtes initiales en langage naturel. Cependant, ces patrons sont spécifiques au vocabulaire utilisé pour décrire la source de données à interroger. Nous expérimentons différentes stratégies pour aligner deux ontologies afin d'y trouver des entités correspondantes décrivant deux sources de données différentes. À partir d'un ensemble d'alignements et des patrons de requêtes initiaux, nous réécrivons ces patrons afin de pouvoir interroger les données décrites en utilisant l'ontologie cible.

Mots-Clés : alignement d'ontologies, patrons de requêtes, langage SPARQL.

Contents

Chapter 1

Introduction

Le Web sémantique est une extension du Web où des données structurées sont publiées et partagées en utilisant des modèles tels que RDF, suivant des vocabulaires spécifiés à l'aide d'ontologies. Avec l'exploitation croissante du Web de données liées, plusieurs organisations mettent à disposition leurs données au format RDF. La manipulation de ces données se fait surtout au travers du langage SPARQL, un langage de requête proposé par le W3C.

Si SPARQL est le langage standard *de facto* pour interroger des données RDF, sa complexité restreint son utilisation à grande échelle, notamment pour les utilisateurs non-spécialistes de RDF. La traduction de requêtes exprimées en langage naturel en requêtes SPARQL fait l'objet de recherches dans les domaines du traitement du langage naturel et du Web sémantique. Dans le système SWIP [Pradel et al., 2012b], les utilisateurs expriment des requêtes en langage naturel et des patrons de requêtes pré-écrits sont instanciés en fonction de l'analyse syntaxique de la requête initiale. Dans [Elbassuoni et al., 2010, Elbassuoni and Blanco, 2011], les requêtes sont limitées à des mots-clés, et un ensemble de sous-graphes, correspondant aux mots-clés de la requête, est classé en fonction de modèles statistiques. Dans [Russell and Smart, 2008], les utilisateurs peuvent exprimer leurs requêtes en utilisant un langage de requête visuel, appelé vSPARQL, qui fournit un formalisme graphique pour spécifier des requêtes SPARQL.

Ce travail se concentre sur les approches fondées sur des patrons de requêtes. L'une des principales limitations de ces approches est que la réutilisation des patrons pour différents ensembles de données n'est possible que dans un cadre très limité. Pour chaque source de données à interroger, les patrons de requêtes doivent être (manuellement) construits, et une bibliothèque de patrons doit être maintenue. Considérant que ces patrons sont définis à partir des ontologies décrivant les sources de données, une stratégie pour automatiser la génération de patrons consiste à exploiter des approches d'alignement d'ontologies. Cette tâche consiste à trouver un alignement (i.e., ensemble de correspondances) entre les entités de deux ontologies (concepts, propriétés ou instances) [Euzenat and Shvaiko, 2007]. Un ensemble de correspondances entre deux ontologies est appelé un alignement. Un alignement peut être utilisé, par exemple, pour générer des expressions qui traduisent automatiquement les instances d'une ontologie sous une ontologie intégrée ou pour la traduction de requêtes d'une ontologie à une autre.

1.1 Objectif

L'objectif principal de ce travail est de proposer un mécanisme automatique pour générer des patrons de requêtes dans le système SWIP. Les premières expérimentations menées dans ce sens sont présentées. À partir d'alignements entre une ontologie source et une ontologie cible, et de patrons de requêtes initiaux, disponibles pour l'ontologie source, nous réécrivons ces patrons afin de pouvoir interroger les données décrites à l'aide de l'ontologie cible. Différentes techniques sont utilisées pour aligner les ontologies d'un jeu de données satisfaisant l'ontologie du Cinéma [Pradel et al., 2012c] et d'autres ontologies sur des domaines similaires. Nous utilisons également des patrons de requêtes qui ont été construits manuellement pour un sous-ensemble de ces ontologies.

1.2 L'équipe MELODI

Mon stage s'est déroulé au sein de l'équipe MELODI[1] de l'Institut de Recherche en Informatique de Toulouse[2], sous la direction de Cassia Trojahn[3], Maître de Conférences, et Ollivier Haemmerlé[4], Professeur des Universités.

Les recherches de l'équipe MELODI sont axées autour de l'**analyse et la formalisation de la langue naturelle** d'une part, et la **représentation et la modélisation des connaissances** d'autre part.

C'est à ce deuxième axe que Cassia Trojahn, Ollivier Haemmerlé et Camille Pradel[5] (doctorant et auteur du système SWIP) sont rattachés. Notamment, le volet Représentation des Connaissances traite des méthodes et des principes de construction d'ontologies; et de la modélisation et de la représentation de connaissances à l'aide de graphes.

1.3 Organisation du rapport

Ce rapport est organisé comme suit :

- Le Chapitre 2 présente les bases de l'alignement d'ontologies, ainsi que les fondations et les technologies du Web sémantique ;
- Le Chapitre 3 présente les principaux travaux sur les systèmes de questions-réponses sur des données RDF, ainsi que les principales approches d'alignement d'ontologies ;
- Le Chapitre 4 présente nos différentes approches de transformation des patrons de requêtes à l'aide des alignements ;
- Le Chapitre 5 présente les expérimentations que nous avons menées pour aligner les différentes ontologies et discute les résultats obtenus.
- Le Chapitre 6 conclut le mémoire et présente les perspectives pour des travaux futurs.

[1] http://www.irit.fr/-Equipe-MELODI-
[2] http://www.irit.fr/
[3] http://www.irit.fr/~Cassia.Trojahn
[4] http://www.irit.fr/~Ollivier.Haemmerle
[5] http://www.irit.fr/-Annuaire-?lang=fr&code=6952

Chapter 2

Contexte

2.1 Alignement d'ontologies

Les progrès des technologies de l'information et de la communication mettent à disposition une quantité énorme d'informations hétérogènes. Si l'hétérogénéité est une caractéristique du Web, le problème de sa gestion entre les différentes sources d'information est croissante. En conséquence, diverses solutions ont été proposées pour en faciliter le traitement, et en particulier, pour automatiser l'intégration de sources de données distribuées. Parmi ces technologies sémantiques se trouve l'alignement d'ontologies. Une ontologie fournit généralement un vocabulaire décrivant un domaine d'intérêt et une spécification du sens des termes utilisés dans le vocabulaire. L'alignement d'ontologies désigne le processus de découverte de correspondances entre les entités sémantiquement liées de deux ontologies diffèrentes [Euzenat and Shvaiko, 2007]. Il désigne également le résultat de ce processus, c'est-à-dire l'expression des correspondances. Ces correspondances peuvent être utilisées pour des tâches variées, allant de la fusion d'ontologies, aux systèmes de questions-réponses, à la conversion ou la traduction de données, ou pour la navigation sur le Web sémantique.

Afin d'illustrer le problème de l'alignement d'ontologies, prenons les deux ontologies simples O_1 et O_2 de la Figure 2.1. Ces deux ontologies décrivent le domaine du cinéma. Les *classes* sont présentées dans des rectangles aux coins arrondis. Par exemple, dans O_1, *FilmParGenre* est une spécialisation (sous-classe) de *Film*. Les *relations*, quant à elles, sont représentées en vert et sans aucun contour. *Titre* est un *attribut* de *Film* défini sur le domaine *String*, et *a produit* est une *propriété* de *Producteur*. *Festival de Cannes* est une instance partagée. Les correspondances sont indiquées par des flèches épaisses reliant une entité de O_1 à une entité de O_2. Elles sont annotées avec le type de relation exprimé par la correspondance: par exemple, *Réalisateur* dans O_1 est moins général ($\sqsubseteq$) que *Person* dans O_2.

Supposons qu'une société de production, de distribution ou d'exploitation cinématographique en acquière une autre. Cette acquisition nécessite l'intégration de leurs sources de données, et par conséquent, des ontologies de ces deux sociétés. Les documents et les données (instances) des deux sociétés sont stockés et organisés selon les ontologies O_1 et O_2, respectivement. Dans cet exemple, ces ontologies contiennent des déclarations de subsomption, des spécifications de propriétés et des descriptions d'instances. La première étape dans l'intégration des ontologies est leur mise en correspondance, pour identifier les concordances, à savoir les entités équivalentes ou ayant des relations de subsomption. Une fois que les correspondances entre les deux ontologies ont été déterminées, elles peuvent être utilisées, par exemple, pour générer des requêtes traduisant automatiquement les instances de ces ontologies dans une ontologie

intégrée. Par exemple, les attributs avec les étiquettes *titre* dans O_1 et *originalTitle* dans O_2 doivent être fusionnés, tandis que la classe *Film* dans O_1 doit être subsumée (englobée) dans la classe *Work* de O_2.

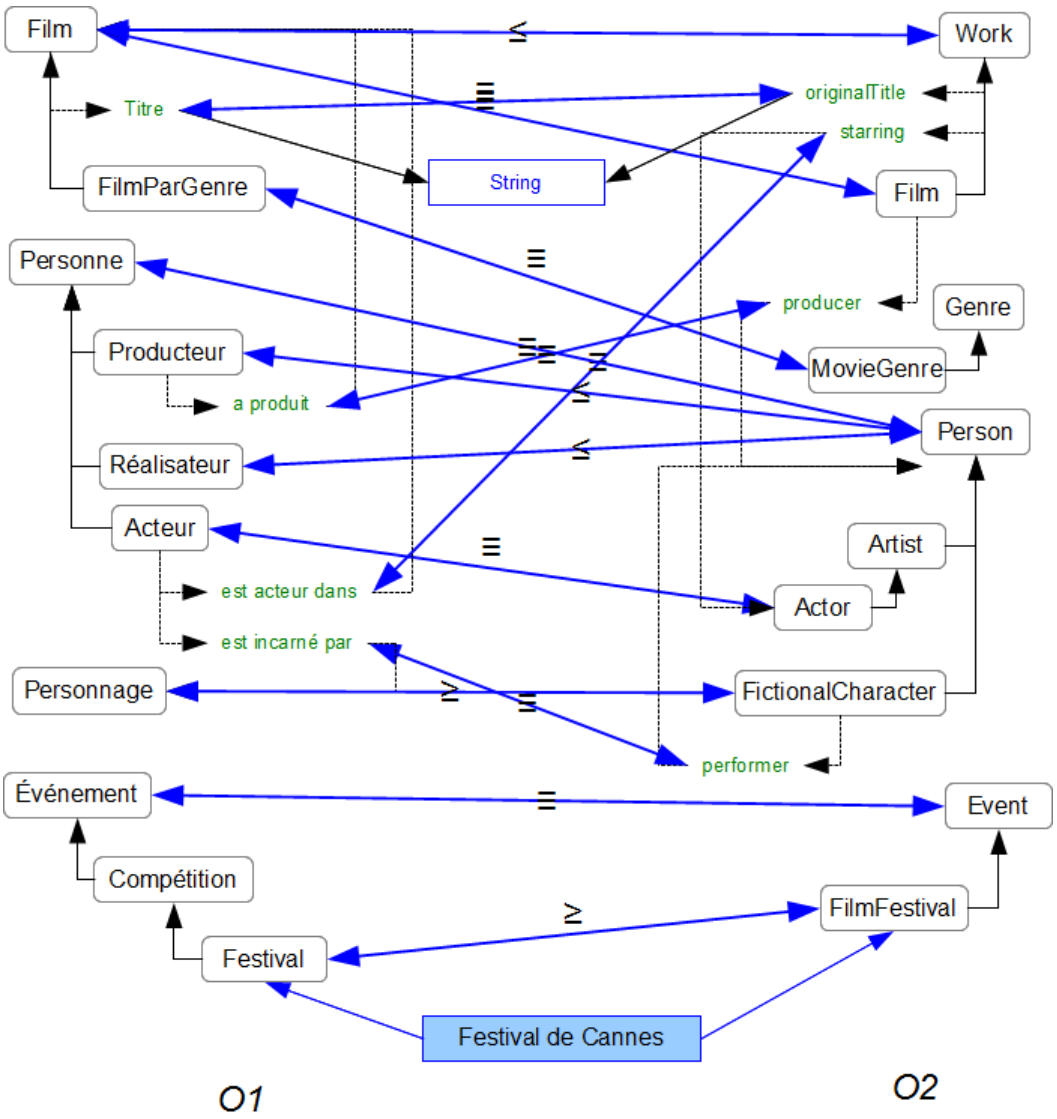

Figure 2.1: Exemple d'alignement entre l'ontologie Cinéma IRIT (O_1) et l'ontologie de DBpedia (O_2).

2.1.1 Définitions

Le processus d'alignement détermine un alignement A pour une paire d'ontologies O_1 et O_2. Le processus peut prendre en compte d'autres paramètres tels que:

- un alignement d'entrée R, qui peut servir d'alignement initial et devant être complété par le processus;
- les paramétres d'entrées (les poids, les seuils, etc.);
- des ressources externes, telles que des thésaurus ou des dictionnaires spécifiques du domaine.

Ce processus génère en sortie un ensemble de correspondances, i.e., un alignement A entre O_1 et O_2 (Figure 2.1.1).

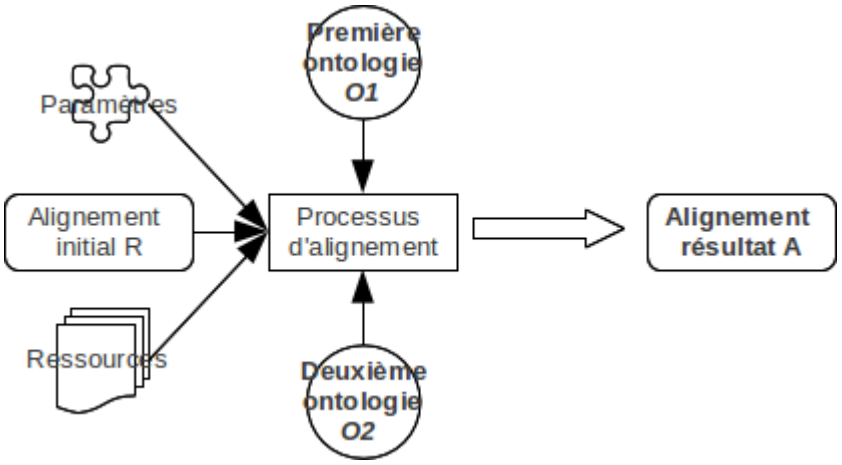

Figure 2.2: L'opération d'alignement

Definition 1 (Alignement) *Un alignement A entre deux ontologies O_1 et O_2 est un ensemble de correspondances $A = \{c_1, c_2, ..., c_n\}$ tel que les c_i sont des couples $\langle {S_i}^1, {S_i}^2 \rangle$, avec ${S_i}^1$ un sous-ensemble d'éléments $\in O_1$, et ${S_i}^2$ un sous-ensemble d'éléments $\in O_2$ (un sous-ensemble peut contenir un seul élément). Les couples $\langle {S_i}^1, {S_i}^2 \rangle$ sont uniques dans A. En revanche, un sous-ensemble d'éléments ${S_i}^j$ peut être présent dans plusieurs correspondances c_i (multiplicité des correspondances). Un alignement A est directionnel et se réfère à une ontologie source et une ontologie cible, noté $A_{O_1 \to O_2}$.*

Les correspondances peuvent être simples ou complexes.

Les correspondances simples

Les correspondances (simples) sont définies dans [Euzenat and Shvaiko, 2007] comme suit:

Definition 2 (Correspondance simple) *Étant donné deux ontologies, une correspondance simple est un quadruplet (4-uple):*

$$\langle e, e', r, n \rangle,$$

tel que:

- e *et* e' *sont des entités (classes, propriétés ou instances) des ontologies source et cible, respectivement ;*
- R *est une relation d'équivalence ($\equiv$), est plus général ($\sqsupseteq$), est plus spécifique ($\sqsubseteq$), ou de disjonction ($\perp$) ;*
- n *est un nombre dans l'intervalle* $[0, 1]$.

La correspondance $\langle e, e', n, r \rangle$ affirme la relation *r* relie les entités e et e', avec une valeur de confiance n. Plus la confiance est élevée, plus la relation est probable. Par exemple, $\langle Film, Work, \sqsubseteq, 0.9 \rangle$, affirme que $Film$ dans O_1 est plus spécifique que $Work$ dans O_2, avec une confiance de 0.9.

Les correspondances complexes

La plupart des outils d'alignement trouvent des correspondances simples entre termes atomiques. Toutefois, les besoins réels nécessitent de trouver des correspondances complexes. En effet, il s'avère que les correspondances simples sont trop limitées pour capturer toutes les relations significatives entre les concepts et les propriétés de deux ontologies connexes. Ainsi, pour de nombreux concepts et propriétés, il n'existe pas d'homologues atomiques, alors qu'il est possible de construire des concepts et des propriétés complexes équivalents.

Definition 3 (Correspondance complexe) *Une correspondance est dite complexe quand au moins une des entités, e ou e', est non atomique.*

Une correspondance complexe est exprimée à l'aide d'un langage formel, tel que la logique du premier ordre. Par exemple,

$$\forall x, Autobiography(x) \equiv Book(x) \wedge author(x, y) \wedge topic(x, y)$$

Dans une première ontologie, nous avons le concept atomique $Autobiography$, tandis que dans la seconde ontologie, nous avons le concept général $Book$, et les propriétés $author$ et $topic$. Une autobiographie est donc définie comme étant un livre dont l'auteur est également le sujet. Un deuxième exemple est :

$$\forall x, Bordeaux\_Wine(x) \equiv Vin(x) \wedge terroir(x, "Aquitaine")$$

Dans une première ontologie, nous avons le concept atomique $Bordeaux\_Wine$ désignant un vin bordelais. Dans la seconde ontologie, nous avons le concept général Vin et la propriété $terroir$. Un $Bordeaux\_Wine$ dans la première ontologie correspond à un Vin dont le $terroir$ est l'$Aquitaine$ dans la deuxième ontologie.

2.1.2 Classification des correspondances

Les alignements peuvent être catégorisés selon différentes dimensions:

- **Simple *vs* complexe**: selon que les correspondances relient une entité à une autre (1-1), ou qu'elles impliquent plusieurs entités dans une formulation d'un système formel ou un langage de représentation des connaissances (1-n, n-1, n-n).

- **Homogène *vs* hétérogène**: selon que les correspondances ne concernent que des entités du même type (par exemple, les classes sont liées uniquement à des classes, les propriétés aux propriétés, les individus aux individus).

- **Type de relation**: la signification d'une relation. Une relation peut être de subsomption, d'équivalence, de [1], partielle (ou toute autre relation définie par le système ou l'utilisateur).

[1] Action de disjoindre, à ne pas confondre avec la disjonction logique pour exprimer un OU

2.1.3 Évaluation

Comme en recherche d'information, l'évaluation des systèmes d'alignement d'ontologies suppose l'utilisation de métriques qui ont pour but de permettre la comparaison des modèles entre eux ou la mise au point de leurs paramètres. La *précision* et le *rappel* sont des mesures d'évaluation ordinaires. Elles sont basées sur la comparaison entre un résultat attendu et le résultat effectif du système d'évaluation. Ces résultats sont considérés comme un ensemble d'éléments, par exemple, les documents à extraire. Étant donné que ces mesures sont couramment utilisées et bien comprises, elles ont été re-utilisées pour l'évaluation de l'alignement d'ontologies.

Dans ce cas, les ensembles de documents sont remplacés par des jeux de correspondances, c'est-à-dire les alignements. L'alignement (*A*) renvoyé par le système d'évaluation est comparé à un alignement de référence (*R*). La précision mesure le rapport de correspondances correctement trouvées (*vraies positives*) sur le nombre total de correspondances retournées (*vraies positives* et *fausses positives*). La précision mesure l'*exactitude* de la méthode et le rappel, quant à lui, mesure le rapport de correspondances correctement trouvées (*vraies positives*) par rapport au nombre total de correspondances attendues (*vraies positives* et *vraies négatives*). C'est une mesure de *complétude* du système.

Definition 4 (Précision et Rappel) *Etant donné un alignement de référence R, la précision d'un alignement résultat A est donné par*

$$P(A,R) = \frac{|R \cap A|}{|A|}$$

et le rappel est donné par

$$R(A,R) = \frac{|R \cap A|}{|R|}$$

2.2 Interrogation de graphes RDF

Aujourd'hui, les grands acteurs du Web (Facebook, Twitter ou Google pour ne citer qu'eux) stockent leurs données dans des silos (des bases de données relationnelles le plus souvent) isolés les uns des autres, et les publient au travers d'API Web, par exemple des services Web de type RESTful[2]. Le W3C (World Wide Web Consortium) vise à surpasser l'interopérabilité des services en favorisant l'intégration des données elles-mêmes. Ainsi, le *Web des données liées* (*Linked Data* en anglais) permet la publication de données structurées sur le Web, en les reliant entre elles pour constituer un réseau global d'informations. Il se repose sur les standards existants du Web, tels que HTTP et URI, pour faciliter la navigation et partager l'information non plus uniquement pour les humains mais également pour les machines amenées à traiter cette information. Cela permet d'interroger automatiquement les données, quel que soit leur lieu de stockage, et sans avoir à les dupliquer.

RDF (Resource Description Framework) fait partie des spécifications du W3C. Il s'agit d'un modèle de graphes destiné à décrire de façon formelle les ressources Web et leurs métadonnées.

Un document structuré en RDF est un ensemble de triplets. Un triplet RDF est une association :

$$(sujet, prédicat, objet)$$

- Le **sujet** représente la ressource à décrire

[2] `http://fr.wikipedia.org/wiki/Representational_State_Transfer`

- Le **prédicat** représente un type de propriété applicable à cette ressource
- L'**objet** représente une donnée ou une autre ressource : c'est la valeur de la propriété.

Le sujet, et l'objet dans le cas où c'est une ressource, peuvent être identifiés par une URI (ou être des nœuds anonymes). Le prédicat est nécessairement identifié par une URI.

Un document RDF correspond ainsi à un graphe orienté et étiqueté, et où chaque triplet correspond à un arc dont le label est le prédicat, le nœud source est le sujet et le nœud cible est l'objet. La sémantique d'un triplet RDF peut être traduite en logique du premier ordre, comme suit:

$$\exists objet, \exists sujet\ tq\ prédicat(objet, sujet)$$

La structure de RDF est extrêmement générique et sert de base à un certain nombre de schémas ou vocabulaires dédiés à des applications spécifiques du Web sémantique. Une partie de ces vocabulaires est spécifiée par le W3C:

- **SKOS (Simple Knowledge Organization System)** est une famille de langages formels permettant une représentation standard des thésaurus, classifications (taxonomies) ou tout autre type de vocabulaire contrôlé et structuré
- **RDFS (RDF Schema)** fournit les méta-concepts nécessaires à la contruction d'ontologies en définissant les concepts de classe, sous-classe, ressource, littéral, propriété, de champs de valeurs et de domaines d'application (*rdfs:Class*, *rdfs:subClassOf*, *rdfs:Property*, *rdfs:domain*, *rdfs:range*, *rdf:type*, etc.)
- **OWL (Web Ontology Language)** est fondé sur la logique de description (sémantique basée sur la logique)[3]. Il permet de décrire des **ontologies**, c'est-à-dire qu'il permet de définir des terminologies (concepts + propriétés) pour décrire des domaines concrets. Un domaine se compose d'instance de concepts.

  Aux méta-concepts déjà présents dans RDFS, OWL ajoute les concepts de classes équivalentes, de propriétés équivalentes, d'égalité de deux ressources, de leurs différences, du contraire, de symétrie et de cardinalité, etc.

RDF/XML est une des nombreuses syntaxes *concrètes*[4], définie par le W3C, pour représenter (sérialiser) un graphe RDF dans un document XML.

Le Listing 2.1 donne un exemple d'une base de données de type RDF écrite en XML, et utilisant le vocabulaire (ontologie) FoaF[5], utilisé pour décrire des personnes et les liens entre elles.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rss="http://purl.org/rss/1.0/"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
    <foaf:Person rdf:about="http://example.net/Chewbacca" rdf:nodeID="Chewbacca"
        >
      <foaf:name>Chewbacca</foaf:name>
```

[3]Les logiques de description peuvent être transposées en logique des prédicats du premier ordre

[4]par opposition à la syntaxe abstraite qui définit la structure des données. On parle aussi de format de sérialisation pour une syntaxe concrète.

[5]FoaF signifie Friend of a Friend (`http://www.foaf-project.org/`)

```
        <foaf:img rdf:resource="http://example.net/Chewbacca.jpg"/>
        <foaf:knows rdf:resource="http://example.net/Han_Solo"/>
    </foaf:Person>
    <foaf:Person rdf:about="http://example.net/Han_Solo" rdf:nodeID="Han_Solo">
        <foaf:name>Han Solo</foaf:name>
          <foaf:title>Captain of the Millennium Falcon</foaf:title>
        <foaf:img rdf:resource="http://example.net/Han_Solo.jpg"/>
        <foaf:knows rdf:resource="http://example.net/Luke_Skywalker"/>
        <foaf:topic_interest>http://DBpedia.org/page/Cuisine</foaf:topic_interest>
    </foaf:Person>
    <foaf:Person rdf:about="http://example.net/Darth_Vader" rdf:nodeID="
        Darth_Vader">
        <foaf:name>Darth Vader</foaf:name>
        <foaf:title>Supreme Commander</foaf:title>
        <foaf:knows rdf:resource="http://example.net/Luke_Skywalker"/>
        <foaf:topic_interest>http://en.wikipedia.org/wiki/Dark_side_(Star_Wars)</
            foaf:topic_interest>
    </foaf:Person>
    <foaf:Person rdf:about="http://example.net/Luke_Skywalker" rdf:nodeID="
        Luke_Skywalker">
        <foaf:name>Luke Skywalker</foaf:name>
        <foaf:title>Jedi Knight</foaf:title>
        <foaf:img rdf:resource="http://example.net/Luke_Skywalker.jpg"/>
        <foaf:topic_interest>http://en.wikipedia.org/wiki/Force_(Star_Wars)</
            foaf:topic_interest>
    </foaf:Person>
    <foaf:Person rdf:about="http://example.net/Princess_Leia" rdf:nodeID="
        Princess_Leia">
        <foaf:name>Princess Leia</foaf:name>
        <foaf:img rdf:resource="http://example.net/Princess_Leia.jpg"/>
        <foaf:knows rdf:resource="http://example.net/Luke_Skywalker"/>
    </foaf:Person>

    <foaf:Group>
        <foaf:name>Galactic Empire</foaf:name>
        <foaf:member rdf:nodeID="Darth_Vader"/>
    </foaf:Group>
    <foaf:Group>
        <foaf:name>Rebel Alliance</foaf:name>
        <foaf:member rdf:nodeID="Chewbacca"/>
        <foaf:member rdf:nodeID="Han_Solo"/>
        <foaf:member rdf:nodeID="Luke_Skywalker"/>
        <foaf:member rdf:nodeID="Princess_Leia"/>
    </foaf:Group>

    <foaf:Image rdf:about="http://example.net/Chewbacca.jpg">
        <dc:description>Chewbacca ID photo</dc:description>
    </foaf:Image>
    <foaf:Image rdf:about="http://example.net/Han_Solo.jpg">
        <dc:description>Han Solo ID photo</dc:description>
    </foaf:Image>
</rdf:RDF>
```

Listing 2.1: Exemple de RDF/XML

SPARQL (SPARQL Protocol and RDF Query Language) est le langage de requêtes recommandé par le W3C pour ajouter, modifier, supprimer mais surtout rechercher des données RDF disponibles à travers Internet. SPARQL a une syntaxe proche de celle du SQL, mais est adapté à la structure spécifique des graphes RDF.
L'énoncé de base d'une requête SPARQL est le triplet RDF (sujet, prédicat, objet). Le calcul d'une requête se fait par appariement de graphe: on cherche tous les sous-graphes correspondant au patron de graphe donné dans la requête. En résultat d'une requête SPARQL, nous pouvons avoir zéro, un ou plusieurs sous-graphes résultats, chacun correspondant à un appariement ou une projection du graphe requête. Ici, nous nous intéresserons seulement aux requêtes de type SELECT, dites interrogatives, qui permettent d'extraire d'un ensemble de graphes RDF une collection de sous-graphes correspondant à un ensemble de ressources vérifiant les conditions définies dans une clause WHERE. Ainsi, SPARQL est capable de rechercher des motifs de graphe (*graph patterns*) obligatoires ou optionnels, ainsi que leurs conjonctions et leurs disjonctions. Les résultats des interrogations SPARQL peuvent être des ensembles de résultats ou des graphes RDF.

Les requêtes sont exécutées depuis un *endpoint* SPARQL. Un endpoint SPARQL permet à des agents (humains ou machines) d'interroger une base de données RDF via le langage SPARQL. Les résultats sont normalement retournés dans un ou plusieurs formats exploitables par les machines. Par conséquent, un endpoint SPARQL est principalement conçu comme une interface de programmation (machine-machine) vers une base de connaissance. La formulation des requêtes et la présentation des résultats lisibles par des humains doivent généralement être mis en œuvre par le logiciel appelant (interface humain-machine).

Voici un exemple de requête SPARQL retournant une liste de personnes et leur description, à partir de la base de connaissance du Listing 2.1:

- La requête SPARQL :

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT DISTINCT ?nom ?titre
WHERE {
        ?personne rdf:type foaf:Person.
        ?personne foaf:name ?nom.
        OPTIONAL { ?personne foaf:title ?titre }
}
```

Listing 2.2: Exemple de requête SPARQL

On remarque la déclaration des *espaces de noms* en début, suivis de la requête proprement dite. Le nom des variables est précédé d'un point d'interrogation ?. La ligne $SELECT$ permet de sélectionner l'ensemble des tuples, ou lignes de variables (nom, titre) correspondant aux contraintes de la clause $WHERE$. La première ligne de la clause $WHERE$ se lit : la variable $personne$ est de type Person au sens de l'ontologie FoaF. La seconde ligne permet de définir la variable nom en tant que propriété $name$ de la variable $personne$. La troisième ligne permet de sélectionner le $titre$ de la $personne$, le cas échéant.

- Le résultat SPARQL, que nous représentons sous forme de tableau:

nom	**titre**
Chewbacca	
Han Solo	Captain of the Millennium Falcon
Darth Vader	Supreme Commander
Luke Skywalker	Jedi Knight
Princess Leia	

Table 2.1: Résultat de la requête SPARQL.

Chapter 3

État de l'art

3.1 Les systèmes de Questions-Réponses

Les systèmes de Questions-Réponses (QR) sont des systèmes qui répondent automatiquement à des questions posées par un humain dans un langage naturel. Ils exploitent des méthodes de recherche d'information (RI) et de traitement automatique du langage naturel (TAL). Quelques exemples de questions issus de QALD[1], qui réalise des campagnes d'évaluation de systèmes de QR:

- *Quelles villes allemandes ont plus de 250000 habitants?*
- *Qui était le successeur de John F. Kennedy?*
- *Quelle est la deuxième plus haute montagne de la Terre?*
- *Donne-moi une liste de tous les trompettistes qui étaient chefs d'orchestre.*
- *Quel est le pilote de Formule 1 qui a gagné le plus de courses?*
- *Est-ce que la nouvelle série de Battlestar Galactica a plus d'épisodes que l'ancienne?*

3.1.1 Fonctionnement des systèmes de réponse aux questions

On peut distinguer les systèmes tirant leurs réponses d'une collection de documents non structurés (documents texte, pages Web, etc.) de ceux utilisant une base de données structurée de connaissances ou d'informations, aussi appelée *base de connaissance*. Ici, nous nous intéressons à la deuxième catégorie de systèmes seulement. Une base de connaissance permet de collecter, d'organiser et de partager des connaissances spécifiques à un ou plusieurs domaines spécialisés donnés, sous une forme exploitable par une machine[2].

Une ontologie peut être utilisée pour définir la structure (de graphe) des données stockées dans la base de connaissance, ainsi que le type des entités et leurs relations. Une autre manière de définir une base de connaissance est donc de dire qu'il s'agit d'une ontologie peuplée par des individus. DBpedia,

[1] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/`

[2] On peut étendre la définition de base de connaissance en disant qu'elle est soit lisible par une machine, soit elle est destinée à un usage humain

par exemple, est une base de connaissance dont les informations sont extraites de Wikipedia[3], et ces informations peuplent l'ontologie de DBpedia.

Bien qu'ils ne soient pas limités à ce type de base de connaissance (c-à-d dont les données peuplent une ontologie), nous portons notre attention sur ces systèmes, qui implémentent de surcroît les formats et les technologies du Web sémantique, normalisés par le W3C. Notamment, les données sont exprimées en RDF, et de ce fait, SPARQL est le langage de requête préconisé pour rechercher des données RDF dans la base de connaissance.

De façon générale, nous utiliserons le terme *requête formelle* ou *requête système* pour qualifier les requêtes formulées dans un langage de requête comme SPARQL, par opposition aux *requêtes utilisateurs* exprimées en langage naturel. Exprimer des requêtes SPARQL suppose de connaître la syntaxe du langage, de savoir ce qu'est un modèle de graphe et, surtout, de connaître l'organisation des données (c-à-d la ou les ontologies) dans la base de connaissance. Plusieurs approches proposent d'aider l'utilisateur dans cette tâche. Les travaux présentés dans [Elbassuoni et al., 2010] visent à étendre le langage SPARQL pour permettre à l'utilisateur de saisir des mots-clefs et des métacaractères (jokers) lorsqu'il ne connaît pas *exactement* le schéma de la base de connaissance. D'autres approches similaires assistent l'utilisateur à formuler ses requêtes. Des interfaces telles que Flint[4] et SparQLed[5] mettent en œuvre des fonctionnalités simples comme la coloration syntaxique ou la complétion automatique.

D'autres approches encore disposent d'un éditeur graphique pour saisir les requêtes, telles que [Athanasis et al., 2004] pour les requêtes RQL[6], [Russell and Smart, 2008] et [Clemmer and Davies, 2011] pour les requêtes SPARQL ou CoGui[7] pour les graphes conceptuels. Ce type d'interface permet de configurer, avec des assistants ou librement, des formulaires et sous-formulaires de saisie, de trier les résultats, limiter leur nombre, les regrouper selon divers critères, ou utiliser différentes fonctions d'aggrégation (SUM, COUNT, AVG).

Si ces outils rendent le travail de formulation moins fastidieux, ils nécessitent toujours de connaître le langage de requête utilisé, et il n'est pas envisageable de demander à l'utilisateur *lambda* de saisir des requêtes dans un langage qu'il ne maîtrise pas. Les travaux menés dans [Ferré and Hermann, 2011], [Ferré and Hermann, 2012], et [Ferré et al., 2011] introduisent le nouveau paradigme de *recherche à facettes basée sur les requêtes*. La recherche à facettes donne aux utilisateurs les moyens de filtrer ou classer une collection de résultats selon un ou plusieurs critères (les facettes). Il n'est donc pas tant question de recherche que de filtrage. Une classification à facettes associe à chaque donnée de l'espace de recherche un certain nombre d'axes explicites de filtrage, ici les mots-clefs issus de la requête elle-même.

D'autres travaux visent à générer de façon automatique ou semi-automatique des requêtes formelles à partir des requêtes formulées par l'utilisateur à l'aide de mots-clefs ou directement en langage naturel. Le système **SWIP** se situe dans cette catégorie de systèmes. L'utilisateur exprime son besoin en information de façon intuitive, sans avoir à connaître le langage de requête sous-jacent ou le formalisme de représentation des connaissances utilisé par le système.

Certains travaux ont déjà proposé de construire des requêtes formelles dans différents langages comme SeREQL [Lei et al., 2006] ou SPARQL [Zhou et al., 2007] [Tran et al., 2009] [Cabrio et al., 2012]. Dans ces systèmes, la construction de la requête formelle nécessite les étapes suivantes:

[3] Wikipedia est aussi une base de connaissance, mais à destination des humains seulement.

[4] `http://openuplabs.tso.co.uk/demos/sparqleditor`

[5] `http://sindicetech.com/sindice-suite/sparqled/`

[6] Famille des langages de requête RDF en général, dont SPARQL fait partie.

[7] `http://www.lirmm.fr/cogui/`

- Déterminer les entités et les individus dans la base de connaissance qui correspondent aux mots-clefs de la requête *utilisateur*. C'est l'**annotation sémantique**, et cette étape est déterminante pour le reste du processus (voir la section suivante).
- Construire l'ensemble des graphes de requête possibles reliant les entités précédemment détectées en explorant la base de connaissance. La multiplicité des annotations possibles implique qu'il n'existe pas un seul graphe de requête.
- Classer les requêtes construites par score décroissant.
- Demander à l'utilisateur de choisir puis valider la requête qui correspond à son besoin.

La mise en oeuvre de ce processus soulève de nombreux problèmes, et les approches existantes tentent d'améliorer les performances globales en optimisant (i) l'étape d'annotation sémantique à l'aide de ressources externes (telles que WordNet ou Wikipedia) [Lei et al., 2006] [Wang et al., 2008], (ii) le mécanisme d'exploration de la base de connaissance pour construire les graphes de requête [Zhou et al., 2007] [Tran et al., 2009], (iii) le classement des requêtes obtenues (ranking) [Wang et al., 2008], et enfin (iv) l'identification des relations à l'aide de modèles textuels [Cabrio et al., 2012]. Une autre méthode est adopté par Autosparql [Lehmann and Bühmann, 2011] : après une interprétation élémentaire de la requête en langage naturel, le système propose un ensemble de réponses et demande à l'utilisateur de distinguer les résultats positifs (corrects) de ceux négatifs, afin d'affiner son interprétation initiale (apprentissage par renforcement).

3.1.2 L'annotation sémantique

L'annotation sémantique est le processus qui fixe l'interprétation d'un contenu textuel en lui associant une sémantique formelle et explicite. On dit qu'on détermine l'*identité sémantique* des données annotées. Dans notre contexte, l'annotation sémantique consiste à étiqueter un mot ou un groupe de mots avec une URI pointant vers une entité de la base de connaissance. Le principal problème de l'annotation sémantique réside dans la multiplicité des annotations possibles. Par exemple, pour annoter une question telle que:

Combien y a-t-il d'habitants à Montpellier?

le système doit être en mesure de choisir la bonne réponse au sein de la multitude de réponses possibles: l'utilisateur fait-il référence à la ville de Montpellier en France ou celle au Québec? Un outil qui utiliserait DBpedia comme base de connaissance doit donc fournir une méthode pour désambiguïser les annotations (Montpellier, http://fr.wikipedia.org/wiki/Montpellier) et (Montpellier, http://fr.wikipedia.org/wiki/Montpellier_(Québec)). Généralement, le système d'annotation calcule un score pour chaque entité potentielle (calcul de similarité ou de probabilité), et c'est le rang des candidats dans une liste qui permet de choisir le lien le plus probable.

3.1.3 Le système SWIP

Dans la catégorie des systèmes de QR construisant des requêtes formelles SPARQL à partir de requêtes en langage naturel, le système SWIP implémente une approche originale fondée sur des patrons de requêtes [Pradel et al., 2011]. Les patrons sont des modèles de requêtes qui sont adaptés aux besoins exprimés par l'utilisateur.

Le principe de SWIP repose sur le postulat que, dans les applications réelles, les requêtes formulées sont généralement des variations de quelques familles de requêtes typiques. Par exemple, les questions *"Dans quels films a joué Sean Connery?"* ou *"Quel est l'acteur principal du Seigneur des Anneaux?"* appartiennent à la famille de requêtes demandant les acteurs qui jouent dans des films. Un autre exemple sont les requêtes demandant les membres d'un groupe musical ou les albums d'un artiste.

L'approche de SWIP vise à améliorer l'efficacité de l'étape de construction des requêtes en utilisant des patrons de requêtes prédéfinis. L'utilisation des patrons évite ainsi d'explorer l'ontologie dans le but de lier les concepts identifiés à partir des mots-clés, puisque les relations potentiellement pertinentes sont déjà présentes dans les patrons. Ainsi, le processus bénéficie des familles de requêtes pré-établies pour lesquelles nous savons qu'un réel besoin en information existe.

Le système SWIP possède une base de connaissance comprenant une ontologie OWL du domaine considéré et des données RDF peuplant cette ontologie. Il dispose également d'une bibliothèque de patrons. L'utilisateur pose une question en langage naturel, laquelle est traduite dans un langage pivot, qui sert d'intermédiaire entre le langage naturel et SPARQL. Le rôle du langage pivot consiste à identifier la structure de la phrase (par exemple, adverbe interogatif + sujet + verbe), et à extraire les mots considérés comme significatifs. Dans la première étape du traitement, la requête ainsi traduite est appariée à des éléments de la base de connaissance (concepts, propriétés ou instances). Des patrons de requêtes sont alors associés à ces éléments. Les différentes associations sont proposées à l'utilisateur sous forme de phrases descriptives en langue naturelle. L'utilisateur sélectionne la phrase correspondant à la requête qu'il souhaite effectivement poser. SWIP peut finalement construire la requête SPARQL voulue, en adaptant le patron de requête considéré.

L'une des principales limitations de SWIP est que la réutilisation des patrons pour différents ensembles de données n'est possible que dans un cadre très limité. Pour chaque source de données à interroger, les patrons de requêtes doivent être (manuellement) construits. Considérant que ces patrons sont définis à partir des ontologies décrivant les sources de données, une stratégie pour automatiser la génération de patrons consiste à exploiter des approches d'alignement d'ontologies.

3.2 Les techniques d'alignement d'ontologies

Depuis une dizaine d'années, différentes approches d'alignement d'ontologies ont émergé dans la littérature [Rahm and Bernstein, 2001, Kalfoglou and Schorlemmer, 2003, Euzenat and Shvaiko, 2007]. Leurs principales différences résident dans la façon dont la connaissance encodée et spécifiée dans chaque ontologie est utilisée lors de l'identification des correspondances entre leurs entités ou leurs structures. Ces approches peuvent être classées selon les éléments exploités dans les ontologies (les labels, la structure, les instances, ou la sémantique), ou en fonction du type de discipline(s) auquel elles appartiennent (statistiques, sémantique, linguistique, apprentissage automatique, etc.).

Une classification générale de ces approches peut être trouvée dans [Rahm and Bernstein, 2001]. Selon cette classification, des méthodes terminologiques comparent lexicalement les chaînes de caractères (tokens ou n-grammes) utilisées pour nommer les entités (les étiquettes et les commentaires des entités), alors que les méthodes sémantiques utilisent des modèles sémantiques (interprétation logique) afin de déterminer si une correspondance existe entre deux entités. D'autres approches considèrent la structure interne des ontologies, comme l'ensemble des valeurs possibles des propriétés (attributs et relations), le fait qu'elles soient transitives et/ou symétriques, ou encore leur structure externe, comme la position

des entités dans chacune des deux hiérarchies. Les instances (ou extensions) peuvent également être comparées en utilisant par exemple l'intersection des ensembles d'instances. En outre, la plupart des systèmes d'alignement ne s'appuient pas sur une méthode unique.

Depuis 2004, l'*Ontology Alignment Evaluation Initiative*[8] (OAEI) organise l'évaluation de systèmes d'alignement d'ontologies [Euzenat et al., 2011]. L'objectif principal d'OAEI est de comparer des systèmes et des algorithmes sur la même base de critères et de permettre à quiconque de tirer des conclusions sur les meilleures stratégies d'alignement. Plusieurs dizaines de systèmes d'alignement, implémentant différentes stratégies, ont été évalués depuis. En 2012, 24 systèmes ont participé à la campagne d'évaluation. Dans cette section, nous nous concentrons sur les approches les plus représentatives utilisées par ces systèmes, car ils ont été utilisés dans le cadre de nos expérimentations (Section 5)[9].

Les systèmes AUTOMS et ASE (une version d'AUTOMS) utilisent des approches terminologiques et statistiques pour générer des correspondances simples (équivalences et subsomptions). Hertuda est également basé sur des approches terminologiques mais en procédant à un pré-traitement des chaînes de caractères (tokenisation). Un autre système, YAM, exploite également différentes techniques (terminologiques, structurelles et similitude entre instances). Le système GOMMA est basé sur la composition et la réutilisation d'alignements, et génère des correspondances de façon distribuée sur plusieurs noeuds de calcul en visant l'efficacité lors d'alignement de larges ontologies. Une étape de pré-traitement, suivie par l'utilisation des techniques terminologiques et structurelles et un ensemble de filtres, est utilisé dans le système HotMatch. Au-delà des approches terminologiques et structurelles (également utilisées par MEDLEY), MaasMatch utilise un ensemble de documents contenant des descriptions textuelles des concepts décrits dans les ontologies. La similitude entre les termes de deux ontologies est mesurée à partir de comparaisons entre les vecteurs de termes des documents.

Une approche logique est proposée dans LogMap et LogMapLt (version lite), où le raisonnement et la réparation des incohérences logiques sont utilisés pour trouver des correspondances entre deux éléments. LogMap interagit en temps réel avec l'utilisateur pendant le processus d'alignement, ce qui est essentiel pour des cas d'utilisation nécessitant des correspondances très précises. En suivant également une approche basée sur le raisonnement, OMR combine le résultat du raisonnement avec des techniques terminologiques (comparaison de chaînes de caractères et utilisation de lexiques tels que WordNet).

En se basant sur un alignement initial, généré à partir de comparaisons lexicales entre entités (concepts et propriétés), Optima recherche l'espace des alignements candidats de façon itérative et adopte l'alignement qui maximise la probabilité de mise en correspondance.

Une approche basée sur les techniques de RI est implémentée dans le système ServOMap (et sa version lite ServOMapL), où une ontologie est considérée comme un corpus de documents à traiter (chaque entité, concepts et relations, est vue comme un document sémantique. Dans cette même optique, WeSeE utilise un moteur de recherche Web pour récupérer des documents Web en lien avec les éléments (labels et commentaires des concepts et propriétés) dans les ontologies. Les résultats de la recherche sont ensuite comparés les uns aux autres (en utilisant la méthode TF-IDF), où plus les résultats sont similaires, plus le score de similitude entre les entités augmente. L'exploitation du Web comme un corpus de document est aussi appliquée dans WikiMatch (ou WMatch), où les inter-liens dans Wikipedia sont utilisés pour trouver des correspondances entres les entités.

[8] `http://oaei.ontologymatching.org/`

[9] La description détaillée de chaque système ainsi que leur URL sont disponibles sur `http://om2012.ontologymatching.org/`

Certains des systèmes décrits ci-dessus sont capables de générer des correspondances entre ontologies décrites dans des langues différentes. ASE utilise l'API Microsoft Translator Bing (ainsi que WeSeE et YAM) pour traduire les descriptions en Anglais. AUTOMSv2 suit une approche similaire, mais réutilise un outil appelé WebTranslator. GOMMA utilise une API de traduction gratuite (MyMemory) pour traduire les labels des concepts non-Anglais en Anglais (les traductions sont considérées comme des synonymes). GOMMA crée un dictionnaire bilingue de façon itérative pour chaque ontologie, qui sera utilisé pour le processus d'alignement. Enfin, Wmatch exploite Wikipedia pour extraire des liens inter-langues pour trouver des correspondances entre les ontologies.

Une limitation de ces systèmes est liée au fait qu'ils génèrent uniquement des correspondances simples. Ce type de correspondance n'est pas toujours suffisant pour décrire la véritable relation entre les entités connexes dans diverses ontologies, et des correspondances complexes sont alors nécessaires [Walshe, 2012]. Ces correspondances sont nécessaires pour toute une variété d'applications, de la réécriture de requêtes [Correndo and Shadbolt, 2011] à l'intégration de connaissances. L'un des rares systèmes capable de générer des correspondances complexes, entre schémas relationnels, est IMap [Dhamankar et al., 2004]. Il suit une approche semi-automatique et utilise différentes techniques pour comparer les attributs des schémas (texte, valeurs numériques, etc.).

Les approches proposées dans la littérature pour détecter les correspondances complexes comprennent les approches basées sur les patrons de correspondances complexes [Scharffe, 2009, Ritze et al., 2009], les patrons de correspondances associés à l'analyse linguistique [Ritze et al., 2010], les régles d'alignement exprimées en logique du premier ordre (FOL) [Qin et al., 2007], et la *raffinement* de correspondances simples [Walshe, 2012]. Dans [Qin et al., 2007], les auteurs proposent une approche itérative qui combine des techniques terminologiques (comparaison de chaînes de caractères des labels des concepts et attributs), la structure interne des ontologies (l'ensemble des domaines et valeurs des propriétés), et l'algorithme de réconciliation des instances des ontologies décrit dans [Dong et al., 2005]. Le résultat de ce processus est ensuite représenté comme un ensemble de règles.

Dans [Scharffe, 2009], un ensemble de patrons de correspondances a été proposé. Les correspondances conditionnelles - où les instances d'un concept dans une ontologie sont liées à un concept dans l'autre ontologie uniquement s'ils ont une valeur particulière pour un attribut donné - en incluant les patrons Classe par Type d'Attribut (CAT), Classe par Valeur d'Attribut (CAV), et Classe par Existence d'Attribut (CAE). Les patrons où la valeur d'un attribut doit être modifiée sont appelés Correspondances de Transformation d'Attributs (ATC). Une approche similaire a également été proposée dans [Ritze et al., 2009]. Le langage EDOAL a été proposé afin de capturer plus précisément les correspondances entre les entités ontologiques hétérogènes [David et al., 2011].

Tandis que la plupart des systèmes d'alignement sont limités à la génération de correspondances simples entre les éléments de deux ontologies, de plus en plus le besoin de correspondances complexes se fait ressentir. Très peu de systèmes proposent de générer ce type de correspondances, et ces systèmes sont généralement évalués avec des jeux de données particuliers. Donc, le développement de systèmes d'alignements complexes et l'évaluation systématique (et pour cela, des alignements de référence sont nécessaires) de ces systèmes sont un vrai manque dans la communauté.

Chapter 4

Approches pour la génération des patrons

Dans ce chapitre, nous abordons les différentes approches que nous avons étudiées pour générer des patrons à partir d'un alignement d'ontologies et d'un patron initial. Pour cela, nous devons tout d'abord définir ce qu'est précisément un patron. Cette approche a comme but la réécriture automatique de patrons de requêtes en exploitant l'alignement des vocabulaires utilisés pour décrire les sources de données.

4.1 Définition d'un patron

Nous avons introduit la notion de patron dans la section précédente 3.1.3. Nous en donnons ici une définition formelle.

Un patron est composé d'un graphe RDF qui est le prototype d'une famille de requêtes typiques. Les sommets de ce graphe désignent des concepts ou des types de données, et les arcs représentent des propriétés. Le graphe est connexe, c'est-à-dire qu'il existe un chemin reliant chaque paire de sommets. Le patron est caractérisé par un sous-ensemble des sommets et arcs de ce graphe, appelés éléments *qualifiants*, et qui peuvent être *instanciés*[1] pendant la construction de la requête finale, conformément aux mots-clefs de la requête [Pradel et al., 2012a].

Des sous-graphes du patron peuvent être optionnels (ils pourront être omis dans la requête SPARQL générée), ou répétables (ils pourront apparaître plusieurs fois dans la requête SPARQL). Chacun de ces *sous-patrons* est caractérisé par une cardinalité minimum et une cardinalité maximum. Une cardinalité minimum de 0 signifie que le sous-patron est optionnel, une cardinalité maximum supérieure à 1 signifie que le sous-patron est répétable. La structure d'un sous-patron est récursive[2]. Notons qu'un sous-patron est relié au reste du graphe par un seul sommet (sommet d'articulation). De cette façon, si un sous-patron optionnel est supprimé du reste du graphe (à l'exception du sommet d'articulation), celui-ci est assuré de rester connexe. De même, si un sous-patron est répété, il sera entièrement dupliqué, à l'exception du sommet d'articulation.

Le patron est également décrit par une phrase en langue naturelle dans laquelle un modèle de sous-chaîne distinct est associé à chacun des sommets qualifiants. Cette phrase descriptive sera elle aussi

[1] On dit par exemple qu'un concept est instancié quand on précise un individu de la base de connaissance comme instance de ce concept

[2] On peut considérer que le patron de plus haut niveau est lui-même un sous-patron.

instanciée de façon à ce qu'elle corresponde exactement à la requête SPARQL finale, et conformément aux étiquettes (labels) des individus ou des valeurs utilisés.

Ci-dessous, la définition formelle d'un patron de requête, telle que définie dans [Pradel et al., 2012a]:

Definition 5 (Patron de requête) *Soit G un graphe et v un sommet de ce graphe ; on note $G\backslash v$ le graphe G privé du sommet v et de tous les arcs incidents à ce sommet.*

Un patron p est un quadruplet (G, Q, SP, S) tel que :

- *G est un graphe RDF connexe qui décrit la structure générale du patron et représente une famille de requêtes;*
- *Q est un sous-ensemble d'éléments de G, appelés éléments qualifiants ; ces éléments sont considérés comme caractéristiques du patron et seront pris en compte lors de l'opération d'association de la requête utilisateur au patron concerné. Un élément qualifiant est soit un sommet (classe ou type de données), soit un arc (propriété d'objet ou propriété de données) de G ;*
- *SP est l'ensemble des sous-patrons sp de p tel que, $\forall sp = (SG, v, card_{min}, card_{max}) \in SP$, on a:*
  - *SG est un sous-graphe de G et v un sommet de SG (et donc de G), tels que $G\backslash v$ est non connexe (v est un sommet d'articulation de G que nous appelons sommet de jonction) et admet $SG\backslash v$ comme composante connexe (i.e. tous les sommets de cette composante connexe appartiennent au graphe du sous-patron) ;*
  - *au moins un sommet ou un arc de SG est qualifiant ;*
  - *$card_{min}, card_{max} \in \mathbb{N}$ et $0 \leq card_{min} \leq card_{max}$; ce sont les cardinalités respectivement minimale et maximale de sp qui permettent de définir les caractères optionnel et répétable de sp.*
- *$S = (s, (sw_1, sw_2, ...sw_n), (w_1, w_2, ...w_m))$ est un template de phrase descriptive dans laquelle n sous-chaînes sw_i correspondent aux n sous-patrons du patron, m sous-chaînes distinctes w_j correspondent aux m éléments qualifiants du patron.*

La Figure 4.1 donne un exemple de patron, inspiré de l'ontologie Music[3]. Ce patron exprime la famille de requêtes demandant le genre et la date de sortie d'une oeuvre musicale au sens large (album, chanson, enregistrement), ainsi que les personnes impliquées dans la réalisation de cette oeuvre. Le patron se compose de trois sous-patrons appelés *genre*, *date* et *person*. Tous ces sous-patrons sont optionnels, et seul le sous-patron *person* est répétable: on considère qu'un travail musical ne peut avoir qu'un seul genre et une seule date de sortie, mais que plusieurs personnes peuvent y participer.

Dans le modèle de phrase descriptive, les parties entre crochets correspondent chacune à un sous-patron et sont indicées par l'identifiant du sous-patron, les sommets qualifiants sont soulignés et indicés par l'élément qualifiant concerné.

Lors de la construction finale de la requête, un élément de patron pourra être instancié (on peut aussi dire spécialisé) selon les règles suivantes:

[3] `http://musicontology.com/`

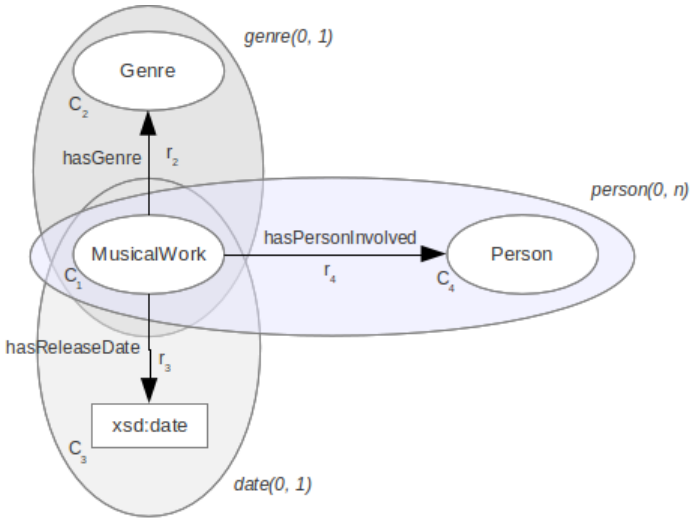

Figure 4.1: A MusicalWork C_1 [of genre $C_2]_{genre}$ [that was released on $C_3]_{date}$ [has for person involvedr_4 a person $C_4]_{person}$

- Une classe c est instanciée avec une sous-classe c' ou une instance de c ;
- Un type de données t est instancié avec une valeur de type t ;
- Une propriété p est instanciée avec une sous-propriété de p.

Le Listing 4.1 présente un patron réel sur l'ontologie Music, dans le format de fichier spécifique à SWIP. Il est composé de 6 sous-patrons délimités par des crochets. Les sous-patrons sont tous articulés autour du concept *mo:MusicalManifestation*. Chaque élément (concept, propriété) est précédé d'un indice qui sera référencé dans la phrase descriptive. Chaque sous-patron est également suivi par son identifiant et ses cardinalités, ainsi que l'indice de l'élément qualifiant du sous-patron.

```
pattern cd_info
 query
 [1_mo:MusicalManifestation 2_foaf:maker(3)     3_mo:MusicArtist;]creator:0..2/3
 [1          6_mo:producer(7)            7_foaf:Agent;       ]producer:0..1/7
 [1          8_mo:composer(9)            9_foaf:Agent;       ]composer:0..1/9
 [1          10_mo:singer(11)            11_mo:Performer;    ]vocal:0..2/11
 [1          12_mo:release_status(13)    13_mo:ReleaseStatus;]status:0..1/13
 [1          14_mo:lyricist(15)          15_mo:MusicArtist;  ]lyricist:0..1/15
 end query
 sentence
   <explanatory sentence here...>
 end sentence
end pattern
```

Listing 4.1: Exemple de patron sur l'ontologie Music (préfixe mo) représentant les questions sur l'édition/production/publication d'une oeuvre musicale.

4.2 Approche basée sur les correspondances simples

Comme première approche, nous avons conçu un transformateur simple de patrons. Le transformateur prend en entrée un alignement, exprimé dans le format d'alignement [David et al., 2011], et un fichier de patrons (voir le Listing 4.1). Il transforme l'ensemble des patrons contenu dans ce fichier en un autre ensemble de patrons, puis le sérialise dans un nouveau fichier en sortie.

Definition 6 *Soit un alignement A et un ensemble de patrons P, un transformateur de patrons peut être défini comme une fonction T telle que:*

$$T(A, P) = P'$$

où P' est le patron généré à partir de A et P. Cette définition reste vraie quelque soit le type d'alignement (correspondances simples ou complexes).

Dans cette première approche, l'alignement ne contient que des correspondances atomiques homogènes (correspondances simples). L'algorithme de la fonction de transformation T est le suivant: pour chaque concept ou propriété e d'un patron donné, nous cherchons dans l'alignement une correspondance contenant l'entité source e. L'entité cible e' dans cette correspondance est ensuite écrite dans le patron de sortie.

Cette approche permet de soulever le problème de la *complétude* des alignements. Un *alignement parfait* ou *alignement complet* - où chaque entité source a un homologue dans l'ontologie cible - est très peu probable. Néanmoins, nous espérons obtenir un *alignement maximal* contenant le plus grand nombre possible de correspondances. Plus précisément, nous espérons que l'ensemble S_p des entités présentes dans le patron p en entrée sera inclus dans l'ensemble S_A des entités sources de l'alignement A, tel que ($S_p \cap S_A = S_p$), du moins que l'intersection de ces deux ensembles sera maximale ($S_p \cap S_A \rightarrow S_p$). On veut donc que chacune des entités des patrons donnés trouve une correspondance dans l'alignement.

Enfin, cette approche pose également le problème de la *granularité* de la transformation: les entités sont traitées une par une, et de façon indépendante. Une propriété peut notamment être remplacée par sa correspondance, sans même considérer la classe des sujets liée à cette propriété (rdfs:domain), et la classe ou le type de données des valeurs de la propriété (rdfs:range). Il faut donc que l'alignement soit correct non seulement pour la propriété, mais aussi pour les deux extrêmités de la relation.

Cette approche a été dirigée par l'expérimentation directe plutôt que par un raisonnement préalable. Elle nous a permis de nous familiariser avec les patrons de SWIP, et les outils constituant l'état de l'art en alignement d'ontologies. Le transformateur est un premier pas vers une solution hybride qui appliquerait d'autres stratégies pour compléter la transformation fondée sur l'alignement, dans le cas où des entités n'auraient pas été remplacées. Nous verrons quelles peuvent être ces stratégies dans le chapitre suivant 5.

Le transformateur a été développé en Java, en s'appuyant sur l'API d'Alignement [David et al., 2011]. Il analyse simplement le fichier de patrons en entrée et *matche* les entités e grâce à des expressions régulières, sans connaître la syntaxe utilisée par SWIP.

Les sections suivantes présentent de nouvelles approches exclusivement fondées sur les alignements d'ontologies avec des correspondances complexes. Pour cela, nous introduisons les clause de Horn avant de décrire l'approche basée sur les correspondances complexes.

4.3 Clauses de Horn

Les correspondances simples, i.e., exprimant des relations entre des entités atomiques, ne sont pas suffisantes pour décrire la relation entre les entités connexes dans diverses ontologies. Des correspondances complexes sont donc nécessaires. Une nouvelle approche pour la réécriture de patrons consiste à utiliser des clauses de Horn, qui sont des correspondances complexes particulières. Des travaux similaires ont été menés dans [**?**].

Definition 7 (Alignement complexe fondé sur les clauses de Horn) *Un alignement complexe entre une ontologie source O_1 et une ontologie cible O_2 est un ensemble de clauses de Horn (strictes):*

$$M_{O_1 \to O_2} = \{M_1, M_2, ..., M_k\}$$

Une correspondance complexe $M_i (i = 1, 2, ..., k)$ est une clause de Horn stricte (comportant un et un seul littéral positif et au moins un littéral négatif), de la forme:

$$O_1 : F \leftarrow O_2 : L_1 \wedge O_2 : L_2 \wedge ... \wedge O_2 : L_n$$

où F et $L_j (j = 1, 2, ..., n)$ sont des littéraux dans O_1 et O_2, respectivement.
Ici, on étend la définition de clause de Horn stricte en disant qu'une correspondance ne peut être qualifiée de complexe que si elle a au moins deux littéraux négatifs $(n \geq 2)$.

On remarque qu'en pratique, on peut intervertir O_1 et O_2 dans la clause de Horn M_i. Le sens de l'implication n'a pas de lien avec le sens de l'alignement. Dans ce cas, l'alignement obtenu est complètement différent. On peut ainsi imaginer un algorithme pour identifier des clauses de Horn en deux passes, et fusionner les deux alignements obtenus.

Rappelons qu'un littéral est un prédicat en logique du premier ordre (FOL) représentant une classe ou une propriété, et pouvant être appliqué sur des constantes ou des variables:

- Une classe correspond à un prédicat unaire avec une seule variable libre (*Actor(X)* ou *MusicalArtist(X)*);
- Une propriété correspond à un prédicat binaire avec deux variables libres (comme *producer(X,Y)*) ;
- Une instance correspond à une constante (*Actor(Robert Redford)*).

Pour illustrer l'intérêt de l'usage des clauses de Horn, prenons deux ontologies O_1 et O_2 sur le domaine du vin. Nous pouvons avoir la correspondance complexe suivante:

$$O_1 : Bordeaux\_Wine(x) \leftarrow O_2 : Vin(x) \wedge O_1 : hasTerroir(x, Aquitaine)$$

où:

- $Bordeaux\_Wine$ et Vin sont des classes dans O_1 et O_2 respectivement ,
- $hasTerroir$ est une propriété dans O_2 dont la classe des sujets (rdfs:domain) est Vin, et la classe des valeurs est $Region$ (rdfs:range) ,
- $Aquitaine$ est une instance de la classe $Region$.

Intuitivement, les clauses de Horn offrent un cadre formel bien fondé, qui préfigure une automatisation *aisée* de leur traitement et leur détection: on peut itérer sur chacune des entités de l'ontologie source et identifier une conjonction d'entités (littéraux) dans l'ontologie cible lui correspondant. De même, on peut itérer sur chacune des entités d'un patron et la remplacer avec la conjonction d'entités correspondantes.

Cependant, les clauses de Horn souffrent d'une limitation fondamentale: elles ne peuvent pas exprimer de disjonction positive. Par exemple, l'affirmation suivante est impossible (dans le cadre des clauses de Horn):

$$O_1 : WineFlavor(x) \leftarrow O_2 : Acidité(x) \vee O_2 : Astringence(x) \vee O_2 : Amertume(x)$$

Malgré cela, nous avons pensé que l'expressivité des clauses de Horn était suffisante pour couvrir de nombreux cas d'alignements d'ontologies. A titre d'expérience, nous avons essayé d'identifier manuellement une conjonction d'entités dans l'ontologie de DBpedia pour chaque entité présente dans les patrons pour Music et Cinema IRIT, formant ainsi des clauses de Horn. Nous n'avons trouvé aucun résultat. Ceci ne prouve pas qu'il n'existe pas d'alignement $A_{Music \rightarrow DBpedia}$ ou $A_{CinémaIRIT \rightarrow DBpedia}$ contenant des clauses de Horn sur l'ensemble des termes des ontologies testées.

Le cadre formel des clauses de Horn semble donc trop rigide pour exprimer de réelles correspondances complexes. Nous devons pouvoir nous affranchir de ces limites et formuler n'importe quel énoncé logique, en respectant tout de même quelques contraintes.

4.4 Correspondances complexes

Dans un premier temps, nous avons construit des alignements complexes à la main, à l'aide des constructeurs des Logiques de Description (DL) [Baader et al., 2003]. Les DL sont des logiques permettant de représenter des connaissances, et de raisonner à partir de ces connaissances. Elles permettent de définir des concepts atomiques et des propriétés (rôles en DL). Elles permettent également de déclarer l'inclusion ($\sqsubseteq$) et l'équivalence ($\equiv$) de concepts ou de rôles (on retrouve les correspondances simples). On peut dès lors avoir des concepts complexes composés de concepts atomiques, et de même pour les rôles.

Par exemple, nous avons les concepts atomiques $Parent$, $Masculin$, $Féminine$, $Mère$, et les rôles atomiques $mèreDe$, $frèreDe$, etc. Nous pouvons déclarer le concept complexe $Femme \equiv Personne \sqcap Féminine$, qui se lit *"Une femme est une personne de sexe féminin"*. Nous pouvons également déclarer que $Mère \equiv \exists mèreDe.\top$ ou $mèreDe$ est un rôle ($\exists mèreDe.\top$ est un concept complexe). Cela signifie qu'être une mère implique une relation avec quelque chose ($\top$) ou encore, être une mère implique d'avoir un enfant. Un autre exemple: "être un oncle" se traduit par $Personne \sqcap (\exists frèreDe(\exists parentDe.\top))$.

Par convention, et pour faire la disctinction avec les rôles, les concepts commencent par une majuscule.

Notons que les DL sont un sous-ensemble de la logique des prédicats du premier ordre (FOL), il est donc possible de transposer des énoncés en DL en FOL; et inversement, mais seulement si l'énoncé logique exprimé en FOL respecte le fragment DL. Ainsi,

$$Personne \sqcap (\exists frèreDe(\exists parentDe.\top))$$

est équivalent à

$$Personne(x) \wedge \exists y(frèreDe(x, y) \wedge \exists z parentDe.(y, z))$$

Pour exprimer des correspondances complexes en DL, la seule contrainte est que les concepts et les rôles (atomiques ou complexes), utilisés à gauche et à droite de l'inclusion ou l'équivalence, doivent

appartenir exclusivement aux ontologies source et cible respectivement. Les constructeurs que nous utiliserons pour contruire de telles correspondances complexes sont A, $\top$, $\bot$, C, $C \sqcap C$, $C \sqcup C$, $\forall R.C$, et $\exists R.C$, avec $A \in ConceptsAtomiques$, $R \in RôlesAtomiques$ et C sont les concepts complexes.

Enfin, pour assurer la sémantique de l'ensemble des correspondances (simples et complexes), il faut que les concepts atomiques, utilisés dans les expressions complexes à gauche et à droite de l'inclusion ou l'équivalence soient d'abord reliés, par des correspondances simples (*principe de consistance*).

Par exemple, si nous voulons aligner deux ontologies O_1 et O_2 sur le domaine de la musique, et que nous déclarons:

$$O_1 : MusicalWork \sqcap O_1 : maker.O_1 : MusicArtist \equiv O_2 : OeuvreMusicale \sqcap O_1 : auteur.O_1 : Artiste$$

Nous devons avoir au préalable les correspondances simples $O_1 : MusicalWork \equiv O_2 : OeuvreMusicale$ et $O_1 : MusicalArtist \sqsubseteq O_2 : Artiste$, sinon la correspondance complexe ci-dessus n'est pas *consistante*.

4.5 Approche généralisée

Nous pouvons combiner les différentes méthodes de transformation des patrons (voir les Sections 4.2, 4.3 et 4.4) à partir d'un alignement contenant des correspondances complexes, et en déduire une approche généralisée. Les correspondances simples sont un fragment (sous-ensemble) des correspondances complexes, de même que les clauses de Horn, exprimées en DL, sont un fragment des correspondances complexes.

L'algorithme 1 itère sur un ensemble de patrons en entrée, et cherche une correspondance dans l'alignement pour chaque sous-patron. S'il n'en trouve pas pour le sous-patron considéré, il itère sur ses sous-éléments (les triplets) et tente à nouveau de trouver une correspondance pour chacun d'entre eux, de façon récursive. A la fin de ce processus, l'algorithme cherche les correspondances pour les éléments (atomiques) restants.

Algorithm 1 Calculate a set of query patterns from an input complex alignment and an input set of query patterns

Require: A: Alignement, P: set of patterns

```
R ← ∅
for p ∈ P do
    SP ← subpatterns of p
    for sp ∈ SP do
        c ← correspondence of sp in A
        if c ≠ nil then
            sp ← c
        else
            for triple ∈ sp do
                c ← correspondence of triple in A
                if c ≠ nil then
                    triple ← c
                else
                    ADD subject, predicate, object to R
                end if
            end for
        end if
    end for
end for
if R ≠ ∅ then
    for r ∈ R do
        c ← correspondence of r in A
        if c ≠ nil then
            r ← c
        end if
    end for
end if
```

Chapter 5

Expérimentations et discussion

5.1 Jeux de données

5.1.1 Ontologies

Nous avons tout d'abord travaillé sur l'ontologie du Cinéma [Pradel et al., 2012c]. Cette ontologie a été développée en parallèle du système SWIP, et des patrons de requêtes l'ayant pour cible sont fournis par défaut. Pour expérimenter des techniques d'alignement en utilisant cette ontologie, nous avons récupéré un ensemble d'ontologies sur des domaines similaires[1] (Tableau 5.1).

Ces ontologies peuvent être classées selon les critères suivants :

- **mono-domaine *vs* multi-domaines**: selon que l'ontologie modélise un domaine spécifique (ontologie de domaine spécifique), représentant une certaine partie du monde, ou si elle expose des concepts sur différents domaines de connaissance[2]

- **monolingues *vs* multilingues**: selon que le nommage des entités est réalisé à l'aide d'un langage naturel unique ou plusieurs

- **présence *vs* absence d'individus**: selon que l'ontologie est peuplée avec des individus (partie ABox des ontologies) ou non.

[1] Certaines de ces ontologies sont disponibles depuis `http://ontologies.alwaysdata.net/cinema/description/`

[2] Attention, ici, une ontologie multi-domaines ne doit pas être confondue avec une ontologie de *haut niveau* (appelée aussi ontologie *fondationnelle*) qui décrit des concepts très généraux qui *devraient* être identiques dans tous les domaines de connaissance.

[3] `http://ontologies.alwaysdata.net/cinema`

[4] `http://www.movieontology.org/2010/01/movieontology.owl`

[5] `http://linkedmdb.org/`

[6] `http://www.jedfilm.com/cinema-ontology`

[7] `http://purl.org/ontology/mo/`

[8] Les métriques données pour l'ontologie Music par exemple inclut les entités des ontologies importées. Une ontologie au format OWL peut ainsi référencer d'autres ontologies OWL contenant d'autres définitions, et dont la signification est considérée comme faisant partie de la signification de l'ontologie qui les importe.

[9] `http://mappings.DBpedia.org/server/ontology/classes/`

[10] `http://purl.org/ontology/po/`

Ontologie	Domaine	Langue	#Classes	#Relations	#Attributs	Bases de connaissance
Cinéma IRIT[3]	Cinéma	Français	198	80	22	-
Movie Ontology[4]	Cinéma	Anglais	78	38	4	LinkedMDB?[5]
JEDFILM[6]	Cinéma	Anglais	343	70	241	-
Music Ontology[7]	Musique	Anglais	91[8]	126	32	LinkedBrainz
DBpedia 3.8 [9]	*Multi*	Anglais	408	821	984	DBpedia
BBC Program[10]	Audiovisuel	Anglais	103	128	80	-

Table 5.1: Le jeu de données.

Les ontologies **Cinéma IRIT** et **Movie** visent à fournir un vocabulaire contrôlé pour décrire sémantiquement des concepts liés aux films, comme *Movie*, *Genre*, *Director*, *Actor*. Cinéma IRIT est la seule ontologie dans notre jeu de données entièrement en français.

Le projet d'ontologie de **JEDFILM** produit également une ontologie sur le domaine du cinéma, décrivant plus largement l'industrie du cinéma, la culture cinématographique comme le visionnage d'un film, les critiques de films, l'étude d'un film, ou encore la sous-culture propre à un ensemble de fans.

L'ontologie **Music** propose des concepts et des propriétés pour décrire la musique (artistes, albums, chansons, mais aussi spectacles, arrangements, etc.). Les constructions qui organisent des connaissances liées à la musique, telles que les classifications (ou taxonomies), sont relativement simples. C'est un bon exemple d'un domaine concret ayant trouvé un accord général. En conséquence, l'ontologie Music est très populaire dans le Web sémantique et est largement adoptée. Le domaine de la musique a également le grand avantage de fournir un grand nombre de données sur Internet: la communauté **MusicBrainz**[11] maintient une large base de données publique (relationnelle) pour la musique, et le projet **LinkedBrainz**[12] publie ces données sous la forme de triplets RDF, peuplant ainsi l'ontologie Music (180 millions de triplets RDF). Le système SWIP fournit également un ensemble de patrons de requêtes visant l'ontologie Music.

L'ontologie de **DBpedia**, quant à elle, est la seule dans notre ensemble de données ayant un large éventail d'entités couvrant différents domaines de connaissance (information géographique, personnes, organisations, films, musique, livres, etc.). Il est noté dans le Tableau 5.1 que l'ontologie DBpedia est en anglais. Si la première version de DBpedia était effectivement exclusivement en anglais, du fait que les informations étaient extraites depuis la version anglophone de Wikipedia seulement, l'ontologie et les dépôts RDF associés ont depuis été enrichis avec d'autres éditions linguistiques, dont une version francophone (par définition, les ressources du Web sémantique ne devraient pas être contraintes par une langue particulière puisqu'elles organisent les données uniquement d'après leur *sens*). Notamment, les classes et les propriétés sont maintenant annotées avec des labels dans différentes langues.

Enfin, l'ontologie **BBC Program** fournit un vocabulaire très simple décrivant les programmes audiovisuels et couvre les séries (saisons), les épisodes, des événements de diffusion, les services de diffusion, etc.

[11] `http://musicbrainz.org/`
[12] `http://linkedbrainz.c4dmpresents.org/`

5.1.2 Les jeux de tests

Pour expérimenter l'alignement d'ontologies, nous avons créé différents jeux de tests. Un jeu de tests consiste en une paire d'ontologies à aligner. Rappelons qu'un alignement A est directionnel et qu'il se réfère à une ontologie *source* S et une ontologie *cible* T, noté $A_{S \to T}$. Dans notre contexte, les patrons de requêtes à transformer sont donc exprimés dans les termes d'une ontologie source.

Nous aurions pu essayer toutes les combinaisons possibles des ontologies décrites dans le Tableau 5.1, mais (i) nous devons privilégier celles pour lesquelles l'ontologie source est visée par les patrons dont nous disposons déjà, et (ii) certaines d'entre elles ne sont pas pertinentes (Cinéma IRIT et Music, par exemple), vu que l'intersection des deux ensembles d'entités est *a priori* très réduite. Un autre exemple est BBC Program qui est en fait construite sur la base d'autres ontologies telles que Music et FOAF. Aligner les ontologies Music et BBC Program serait donc équivalent à aligner l'ontologie Music avec elle-même.

D'autre part, nous pouvons aligner une ontologie d'un domaine particulier avec une ontologie multi-domaines (Music-DBpedia), et nous attendre à obtenir une bonne couverture (c-à-d qu'un grand nombre d'entités sources trouve une correspondance). Par conséquent, DBpedia est une bonne candidate pour être une ontologie cible.

Selon ces critères, les combinaisons retenues sont donc les suivantes:

→	**Cinéma IRIT**	**Movie**	**JEDFILM**	**Music**	**DBpedia**	**BBC Program**
Cinéma IRIT		×	×		×	
Movie						
JEDFILM						
Music					×	
DBpedia						
BBC Program						

Table 5.2: Jeux de tests.

Nous présentons les résultats d'alignements sur ces quatre paires d'ontologies dans le Tableau 5.3.

5.2 Systèmes d'alignement

Les systèmes[13] d'alignement que nous avons utilisé pour nos tests ont été présentés dans le Chapitre 3.2. Ces systèmes ont participé à l'*Ontology Alignment Evaluation Initiative* (OAEI) 2012 [Aguirre et al., 2012], ainsi qu'aux campagnes précédentes (Tableau 5.3). Ces systèmes sont pour la plupart libres et accessibles au public.

Pour manipuler les outils d'alignements, il a d'abord fallu se familiariser avec d'autres outils, parmi lesquels:

- La plate-forme **SEALS**[14], dans le cadre du projet SEALS, dédiée à l'évaluation des technologies du Web sémantique. En particulier, l'OAEI et le projet SEALS sont étroitement liés dans le domaine de l'alignement d'ontologies. La plate-forme est prévue pour être utilisée en ligne[15]: elle permet

[13] La description detaillée de chaque système ainsi que son URL sont disponibles sur `http://om2012.ontologymatching.org/`

[14] `http://www.seals-project.eu/`

[15] En pratique, nous l'avons utilisée localement sur notre machine

d'héberger une suite d'outils d'alignement, et de les exécuter en cascade sur des jeux de tests téléchargés au préalable.

- l'**API d'Alignement**[16], utilisée par tous les systèmes que nous avons testés, est une API mais également une implémentation pour exprimer et partager des alignements d'ontologies. L'API permet de produire un format de données pour représenter un ensemble de correspondances entre les entités de deux ontologies (classes, objets, propriétés) de manière uniforme. L'objectif initial est de faciliter l'échange automatisé d'alignements d'ontologies entre systèmes d'informations hétérogènes (interopérabilité). Le format est exprimé en RDF et est extensible.

- L'**API OWL**[17], également utilisée par tous les systèmes d'alignement. Il s'agit d'une API Java et d'une implémentation de référence pour créer, manipuler et sérialiser des ontologies au format OWL. L'API OWL comprend notamment des *parsers* et des *writers* pour manipuler les ontologies d'entrée dans le format RDF/XML ou OWL/XML.

5.3 Résultats et discussion

Le Tableau 5.3 donne le nombre de correspondances trouvées par chacun des systèmes pour les paires d'ontologies considérées.

Nous pouvons distinguer les outils capables de produire des alignements multi-langues de ceux générant des alignements mono-langues. 6 des 24 systèmes appliquent des stratégies spécifiques pour faire face aux ontologies décrites en différentes langues naturelles.

Pour donner une idée du nombre total d'entités à apparier, les entêtes de colonne précisent entre parenthèses le nombre de classes et de propriétés pour chaque ontologie dans la paire considérée.

L'objectif est d'utiliser ces alignements pour la transformation de patrons, et ainsi de couvrir autant d'entités que possible dans l'ontologie source.

Pour obtenir un seul ensemble de correspondances uniques (c-à-d tous les couples $(entité\ source, entité\ cible)$ uniques), nous avons fusionné[18] tous les alignements générés par les systèmes listés dans le Tableau 5.3, pour chaque paire d'ontologies. L'alignement obtenu est principalement caractérisé par sa cardinalité 1-n, où une entité de l'ontologie source peut correspondre à une ou plusieurs entités de l'ontologie cible.

Les meilleurs résultats sont obtenus entre l'ontologie Music et DBpedia. Pour la paire Music-DBpedia, sur 207 correspondances, 168 entités distinctes dans l'ontologie Music (qui en compte 249 au total) ont au moins une correspondance dans DBpedia, ce qui donne une couverture de 67% de l'ontologie source.

Ce résultat est a priori satisfaisant. Cependant, l'ontologie Music utilise d'autres vocabulaires décrivant des concepts plus généraux, tels que FoaF et l'ontologie Event[19]. Nous constatons que seulement 39% des entités qui font la spécificité de l'ontologie Music (c-à-d sans les entités des ontologies externes/importées) trouvent une correspondance dans DBpedia (Tableau **??**).

Pour la paire Cinéma IRIT-DBpedia, seulement 10% des entités sources ont une correspondance dans DBpedia. Nous pouvons noter dans ce cas que le nombre de correspondances est pratiquement le même pour tous les outils. Cinéma IRIT étant en français et DBpedia en anglais, nous nous attendions à ce

[16] `http://alignapi.gforge.inria.fr/`
[17] `http://owlapi.sourceforge.net/`
[18] à l'aide de l'API d'Alignement
[19] `http://motools.sourceforge.net/event/event.html`

	Système	Music → DBpedia (249 * 2213)	Cinéma IRIT → DBpedia (300 * 2213)	Cinéma IRIT → JEDFILM (300 * 654)	Cinéma IRIT → Movie (300 * 120)
Multilangue	YAM	×	**13**	OME	OME
	WeSeE	**31**	IL	IL	IL
	GOMMA	**16**	**13**	**4**	**1**
	AUTOMSv2	×	**6**	NPE	NPE
	Wmatch	**59**	NPE	NPE	NPE
	MEDLEY	**0**	NPE	NPE	NPE
Non spécifique	Aflood	**58**		-	-
	Rimon			-	-
	Falcon	**59**		-	-
	Lily			-	-
	CIDER	x		-	-
	AROMA	**0**	**12**	-	-
	ASE	x		-	-
	GO2A	x	**17**	-	-
	Hertuda	**45**		-	-
	HotMatch	x		-	-
	LogMap	**21**	**17**	-	-
	LogMapLt	**24**	**14**	-	-
	MaasMtch	x		-	-
	MapSSS	x		-	-
	OMR			-	-
	Optima	x		-	-
	ServOMap	x		-	-
	ServOMapL	x	**12**	-	-
Merge	**# correspondances**	**207**	**31**	**4**	**1**
	# entités sources distinctes	**168**	**31**	**4**	**1**
	Couverture onto. source	**67%**	**10%**	**1%**	**1%**

Table 5.3: Les alignements obtenus. Le tableau donne le nombre de correspondances trouvées par chacun des systèmes pour les paires d'ontologies considérées. Une cellule vide indique que le système n'a pas généré d'alignement. Les entêtes de colonne précisent entre parenthèses le nombre de classes et de propriétés pour chaque ontologie dans la paire. NPE: NullPointerException, IL: Infinite Loop, OME: OutOfMemoryException, x: Unknown Alignment Error, -: Non testé

que les outils d'alignement multilangues surpassent ceux qui ne sont pas conçus dans ce sens ; ce n'est pourtant pas le comportement observé. Mis à part des erreurs internes, ces derniers ont en effet trouvé des correspondances, grâce aux similitudes de structure dans les graphes des ontologies.

Enfin, pour les paires Cinéma IRIT-JEDFILM et Cinéma IRIT-Movie, la couverture est réduite à 1%. Ces résultats peuvent s'expliquer par les problèmes rencontrés à l'exécution des outils (fuites mémoires, boucles infinies, NullPointerExceptions). De plus, beaucoup d'entre eux utilisent une ancienne version de OWL API qui n'est pas capable de lire les ontologies en entrée, notamment à cause d'imports d'ontologies externes, et alors qu'une version plus récente de l'API fonctionne correctement sur ces ontologies. Nous avons donc tenté de remplacer l'API avec une version plus récente, mais n'étant pas rétro-compatible, les outils ne marchent plus du tout.

		Music → DBpedia **(Whole)** (249 * 2213)	Music → DBpedia **(Core)** (249 * 2213)
Merge	**# correspondances**	**207**	**119**
	# entités sources distinctes	**168**	**97**
	Couverture onto. source	**67%**	**39%**

Table 5.4: Zoom sur les résultats d'alignements entre Music et DBpedia

L'ontologie Cinéma IRIT n'a pas été alignée auparavant avec d'autres ontologies, et des alignements de référence pour cette ontologie ne sont donc pas disponibles.
Le seul ensemble d'alignements de référence que nous avons trouvé contient les correspondances entre les ontologies Music et DBpedia. Ces correspondances ont été réalisées à la main par des experts et ont été initialement utilisées pour tester la qualité des alignements générés par le système BLOOMS[20]. Etonnamment, l'alignement ne contient que des correspondances entre concepts et ignore leurs propriétés, avec en outre seulement 6 relations d'équivalence contre 639 relations de subsomption. De plus, même si l'équivalence a déjà été établie entre deux concepts, les experts ont identifié toutes les relations de subsomption possibles entre les concepts des deux ontologies (Ex: $Work \equiv Work$, mais également $Work \geq Newspaper$, $Work \geq Book$, $Work \geq Software$, $Work \geq MusicalWork$, etc.). De ce fait, l'alignement de référence contient beaucoup de correspondances, mais avec une très forte multiplicité (1-n). En raison de cette redondance et du fait que l'alignement ne se concentre que sur les concepts, nous ne sommes pas en mesure d'évaluer la qualité des alignements générés conformément à l'alignement de référence: le calcul du rappel et de la précision s'en trouveraient faussés dès le départ.

5.3.1 Résultats de la transformation de patrons

SWIP offre deux ensembles de patrons de requêtes: un pour l'ontologie Music et un autre pour l'ontologie Cinéma IRIT. À partir de l'alignement Music-DBpedia (Tableau 5.3), nous avons transformé les patrons de requêtes pour l'ontologie Music en de nouveaux patrons pour DBpedia. Nous avons obtenu le résultat suivant: 25 entités sur 60 ont été remplacées, soit 41%. Sur un total de 5 patrons, contenant au total 27 sous-patrons, seulement 2 ont été traduits complètement.

Concernant les patrons pour Cinéma IRIT, et en utilisant l'alignement Cinéma IRIT - DBpedia, la couverture de la transformation est proche de zéro. Cela signifie que non seulement l'alignement est de mauvaise qualité, mais aussi que les entités sources dans les correspondances trouvées ne sont pas celles utilisées dans les patrons.

Quel que soit le système d'alignement utilisé, l'ontologie source n'est jamais complètement couverte, et des entités peuvent ne pas avoir d'équivalent dans l'ontologie cible. Ce résultat était attendu et met en évidence le problème de la complétude du processus de transformation. Nous avions néanmoins besoin d'évaluer très précisément cette insuffisance.

Nous pensions que les correspondances simples permettraient de traduire une grande partie d'un patron de requêtes. Nous aurions ensuite appliqué d'autres méthodes plus sophistiquées (ou au contraire

[20] `http://knoesis.org/projects/BLOOMS#Resources_for_Download`

des méthodes dites de force brute) pour transformer la partie restante. Par exemple, si une propriété de l'ontologie source ne peut pas être déterminée dans la cible alors que nous avons trouvé des concepts équivalents pour les deux extrémités de la relation, nous pourrions explorer toutes les relations possibles entre les deux classes correspondantes.

D'autre part, le degré d'appariement entre les ontologies dépend avant tout de leur domaine. En général, on suppose que les deux ontologies à aligner décrivent le même domaine, sinon elles ne se recouvrent pas ou à peine. En outre, les systèmes d'alignement ne doivent pas faire en sorte de trouver une correspondance pour chaque élément des deux ontologies, et ignorer les éléments qui ne sont pas pertinents à la place.En conséquence, l'incomplétude des alignements est tout à fait normale et nous devons savoir la gérer.

Outre le problème d'incomplétude des alignements, la plupart des systèmes produisent des correspondances de qualité: bien qu'elles ne soient pas nombreuses, les correspondances trouvées se révèlent en revanche *vraies* (selon nos propres jugements de pertinence). Mais les outils se limitent à la détection de correspondances simples, avec des relations d'équivalence ou de subsomption entre des entités uniques. Et les correspondances simples ne sont souvent pas suffisantes pour représenter correctement la relation entre les entités alignées.

Il y a aussi le problème majeur de la granularité des correspondances: remplacer les entités une par une dans un patron de requêtes ne permet pas d'obtenir la sémantique attendue résidant dans le graphe dans son ensemble. Notamment, le choix d'une correspondance plutôt qu'une autre avec la même classe source devra se faire en considérant les propriétés utilisées sur cette classe. Nous avons besoin de changer d'échelle, et nous devons être en mesure d'exprimer tout sous-ensemble du graphe. La sémantique réelle d'une classe, et donc sa correspondance, se révèle dans ses relations avec les autres classes. Cependant, les alignements atomiques et homogènes ont une sémantique bien définie, et sont les blocs de construction élémentaires pour obtenir des alignements plus riches, comme nous allons en discuter dans la section suivante.

Enfin, concernant l'outil de transformation que nous avons développé, si une entité source e appartient à plusieurs correspondances dans l'alignement (multiplicité des correspondances), l'outil demande à l'utilisateur de sélectionner l'entité cible e' la plus appropriée. Un processus complètement automatique choisirait la correspondance avec la valeur de confiance la plus élevée, mais (i) l'alignement en entrée peut être le résultat de la fusion de plusieurs alignements, et (ii) la justification et l'attribution de cette valeur ne sont pas normalisées et on constate qu'elles sont inconsistantes sur l'ensemble des systèmes d'alignement testés.

5.4 Nos correspondances complexes

Nos expérimentations initiales ont montré que les correspondances simples, i.e., exprimant des relations entre des entités atomiques, ne sont pas suffisantes pour décrire la relation entre les entités connexes dans diverses ontologies. Des correspondances complexes sont donc nécessaires [Walshe, 2012]. Nous envisageons d'utiliser ce type de correspondances pour la réécriture de patrons de requêtes.

En s'appuyant sur les constructeurs des logiques de description (DL) et les alignements simples entre les ontologies, nous avons construit à la main des alignements complexes qui relient l'ontologie Music et l'ontologie de DBpedia d'une part ($A_{Music \rightarrow DBpedia}$), et l'ontologie Cinéma IRIT et l'ontologie de

DBpedia d'autre part ($A_{CinémaIRIT \rightarrow DBpedia}$). Ces alignements sont preséntés dans les Tableaux 5.5 et 5.7, respectivement.

Ces alignements ont été réalisés à titre d'expérimentation, avec une étendue limitée aux seules entités sources présentes dans les patrons pour Music et Cinéma IRIT. En d'autres termes, la *portée* d'un alignement se limite au *scope* du patron considéré.

Pour la création des alignements, nous avons découpé les patrons en sous-graphes indépendants. Un sous-graphe correspond en général à un sous-patron ou à un élément qualifiant. Nous avons ensuite exprimé chacun de ces sous-graphes en DL. Un sous-graphe est donc soit un concept atomique A, soit un concept complexe de la forme $C \sqcap \exists R.C'$, où C' peut lui-même être de la forme $C \sqcap \exists R.C'$, de façon récursive.
Chacun des sous-graphes ainsi obtenus, exprimé en DL, est l'expression complexe source pour laquelle nous voulons trouver une correspondance dans l'ontologie cible.

Concernant le patron pour l'ontologie Music, nous avons identifié 23 sous-graphes, et nous avons trouvé une correspondance pour 20 d'entre eux dans l'ontologie de DBpedia (87% de couverture). Notamment, les concepts $ReleaseStatus$ (le statut d'une oeuvre musicale ou du support physique d'une oeuvre musicale. Ex: le *bootleg*[21]) et $Record$ (enregistrement sonore) n'ont pas d'équivalent dans DBpedia (voir les correspondances #11, #12 et #24 dans le Tableau 5.5).

Pour le patron Cinéma IRIT, nous avons 51 sous-graphes, et nous avons pu lier 45 d'entre eux à des concepts et des rôles de DBpedia (88% de couverture).

Un rôle se traduit souvent par une composition de rôles (conjonction et/ou disjonction) dans l'ontologie cible. Notez que la disjonction n'est actuellement pas prise en charge dans les patrons de requêtes. Par exemple, la correspondance suivante illustre bien la composition de rôles (#6 dans le Tableau 5.7)):

$$Artiste \sqcap \exists estRecompenseA.CesarDuCinema \equiv Artist \sqcap \exists cesarAward(Award \sqcap \exists event.FilmFestival)$$

De plus, DBpedia ne fournit pas systématiquement des relations inverses, et nous devons parfois introduire l'inverse d'un rôle R, noté R^-, comme dans la correspondance suivante: (#9 dans le Tableau 5.7):

$$Artiste \sqcap \exists acteurDoublePar.Doubleur \equiv Person \sqcap \exists dubber^-.Artist$$

Un cas intéressant est celui de la correspondance #19 dans le Tableau 5.7:

$$Artiste \sqcap \exists estRecompensePour.Film \sqsubseteq Artist \sqcap \exists nominee^-.Film$$

Il y a aussi des cas où nous avons besoin de plus d'expressivité, et utiliser la logique FOL pour exprimer des contraintes sur des variables (#16 dans le Tableau 5.7)

$$\forall x FilmLocalise(x) \rightarrow \exists yz Film(x) \sqcap originalLanguage(x, y) \sqcap language(x, z) \sqcap y \neq z$$

Un film localisé (au sens de régionalisation[22]) est un film traduit dans une langue étrangère. Ceci équivaut dans DBpedia à un film dont la langue est différente de sa langue originale.

Avec ces alignements, nous avons traduit les patrons pour Cinéma IRIT et Music dans les termes de l'ontologie de DBpedia. Nous avons maintenant à disposition deux nouveaux ensembles de patrons pour DBpedia, exprimant des familles de requêtes sur les domaines de la musique et du cinéma. A

[21] `http://fr.wikipedia.org/wiki/Bootleg_(musique)`
[22] `http://fr.wikipedia.org/wiki/Localisation_(informatique)`

titre d'expérimentation, nous avons déployé celui sur la musique dans le système SWIP. La base de connaissance de DBpedia a été chargée dans le système SWIP en vue de sa prochaine participation au *challenge* QALD[23].

Nous avons posé un ensemble de questions au système SWIP relatives à la musique (en faisant attention à ne pas exprimer de requêtes relatives aux trois sous-patrons qui n'ont pas trouvé de correspondance). Les résultats obtenus ne sont pas bons du fait de l'annotation sémantique, qui échoue à trouver les entités et les individus dans la base de connaissance qui correspondent aux mots-clefs de la requête. Le système SWIP est pour l'instant paramétré de telle façon que l'étape d'annotation sémantique n'est efficace que pour la base de connaissance de Music.
La Figure 5.6 illustre ce que pourrait être la construction finale d'une requête SPARQL sur la base de connaissance de DBpedia, en utilisant le nouveau patron, et en comparaison de la même requête sur Music.

Enfin, les tableaux 5.8 et 5.9 donne les alignements complexes entre l'ontologie Music et l'ontologie de DBpedia, obtenus avec l'outil ComplexMapping[Ritze et al., 2009], et avec deux alignements initiaux différents. Les résultats sont peu nombreux et ne sont pas pertinents. A titre de comparaison, ces alignements n'ont aucune correspondance en commun avec les alignements construits à la main.

[23] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/`

1	**Music**		**DBpedia**
2	mo:MusicalManifestation	$\sqsubseteq$	MusicalWork
3	mo:MusicalWork	$\equiv$	MusicalWork
4	MusicArtist	$\equiv$	MusicalArtist
5	MusicManifestation $\sqcap$ $\exists$foaf:maker.MusicArtist	$\equiv$	MusicalWork $\sqcap$ ($\exists$artist.owl:Thing $\sqcup$ $\exists$author.Person $\sqcup$ $\exists$creator.Person $\sqcup$ $\exists$musicComposer.MusicalArtist)
6	MusicalManifestation $\sqcap$ $\exists$producer.foaf:Agent	$\equiv$	MusicalWork $\sqcap$ ($\exists$producer.Person $\sqcup$ $\exists$productionCompany.Company)
7	MusicalWork $\sqcap$ $\exists$produced_work(Composition $\sqcap$ $\exists$composer.foaf:Agent)	$\equiv$	MusicalWork $\sqcap$ ($\exists$musicComposer.MusicalArtist $\sqcup$ $\exists$composer.Person)
8	Performance	$\sqsubseteq$	Event
9	MusicalWork $\sqcap$ $\exists$performed_in(Performance $\sqcap$ $\exists$singer.foaf:Agent)	$\equiv$	MusicalWork $\sqcap$ $\exists$event(Event $\sqcap$ ($\exists$associatedMusicalArtist.MusicalArtist $\sqcup$ $\exists$associatedBand.Band))
10	Performer	$\equiv$	MusicalArtist
11	MusicalManifestation $\sqcap$ $\exists$release_status.ReleaseStatus		N/A
12	ReleaseStatus		N/A
13	MusicalManifestation $\sqcap$ $\exists$lyricist.MusicArtist	$\equiv$	MusicalWork $\sqcap$ $\exists$writer.Person
14	foaf:Group	$>$	Band
15	foaf:Agent $\sqcap$ $\exists$member_of.foaf:Group	$\equiv$	Band $\sqcap$ ($\exists$bandMember.Person $\sqcup$ $\exists$formerBandMember.Person)
16	Membership $\sqcap$ ($\exists$event:agent.foaf:Agent $\sqcap$ $\exists$event:time.(event:TemporalEntity $\sqcap$ $\exists$tl:start.xsd:date)	$\equiv$	Event $\sqcap$ ($\exists$pastMember.Person $\sqcap$ $\exists$startDate.xsd:date)
17	Membership $\sqcap$ ($\exists$event:agent.foaf:Agent $\sqcap$ $\exists$event:time.(event:TemporalEntity $\sqcap$ $\exists$tl:end.xsd:date)	$\equiv$	Event $\sqcap$ ($\exists$pastMember.Person $\sqcap$ $\exists$endDate.xsd:date)
18	MusicGroup	$\equiv$	Band
19	MusicGroup $\sqcap$ $\exists$bio:event(bio:Birth $\sqcap$ $\exists$bio:date.xsd:dateTime)	$\equiv$	Band $\sqcap$ ($\exists$formationDate.xsd:date $\sqcup$ $\exists$formationYear.xsd:gYear)
20	MusicGroup $\sqcap$ $\exists$bio:event.(bio:Death $\sqcap$ $\exists$bio:date.xsd:dateTime)	$\equiv$	Band $\sqcap$ ($\exists$extinctionDate.xsd:date) $\sqcup$ $\exists$extinctionYear.xsd:gYear)
21	foaf:Agent $\sqcap$ $\exists$rel:spouseOf.foaf:Agent	$\equiv$	Person $\sqcap$ $\exists$spouse.Person
22	foaf:Agent $\sqcap$ $\exists$collaborated_with.foaf:Agent	$\equiv$	(Artist $\sqcap$ $\exists$associatedAct.Artist) $\sqcup$ (Person $\sqcap$ $\exists$partner.Person)
23	Track $\sqcap$ $\exists$duration.xsd:decimal	$\equiv$	MusicalWork $\sqcap$ $\exists$runtime.Time
24	Record		N/A

Table 5.5: Correspondances complexes réalisées à la main entre Music et DBpedia

PREFIX **mo: <http://purl.org/ontology/mo/>**	PREFIX **dbo: <http://dbpedia.org/ontology/>**
SELECT DISTINCT ?album WHERE {	SELECT DISTINCT ?album WHERE {
?album a **mo:Record** .	?album a **dbo:MusicalWork** .
?album **foaf:maker** ?artist .	?album **dbo:artist** ?artist .
?artist a **mo:MusicArtist** .	?artist a **dbo:MusicalArtist** .
?artist rdfs:label ?artistLabel .	?artist rdfs:label ?artistLabel .
filter(?artistLabel="Michael Jackson"@en).	filter(?artistLabel="Michael Jackson"@en).
?album **mo:producer** ?producer .	?album **dbo:producer** ?producer.
?producer rdfs:label ?producerLabel .	?producer rdfs:label ?producerLabel.
filter(?producerLabel="Quincy Jones"@en).	filter(?producerLabel="Quincy Jones"@en).
}	}
ORDER BY ?album	ORDER BY ?album

Table 5.6: Requêtes SPARQL alignées, demandant la liste des albums de Michael Jackson produits par Quincy Jones, pour Music et DBpedia.

1	**Cinema IRIT**		**DBpedia**
2	Artiste	$\sqsupseteq$	Actor
3	Artiste	$\sqsupseteq$	Comedian
4	Artiste	$\equiv$	Artist
5	CesarDuCinema	$\sqsubseteq$	FilmFestival
6	Artiste $\sqcap$ $\exists$estRecompenseA.CesarDuCinema	$\equiv$	Artist $\sqcap$ $\exists$cesarAward(Award $\sqcap$ $\exists$event.FilmFestival)
7	Oeuvre	$\equiv$	Work
8	Artiste $\sqcap$ $\exists$contribueArtistiquementAOeuvre.Oeuvre	$\sqsubseteq$	Person $\sqcap$ $\exists$associate$^-$.Work
9	Artiste $\sqcap$ $\exists$acteurDoublePar.Doubleur	$\equiv$	Person $\sqcap$ $\exists$dubber$^-$.Artist
10	Role	$\sqsupseteq$	FictionalCharacter
11	Artiste $\sqcap$ $\exists$incarne.Role	$\equiv$	Person $\sqcap$ $\exists$portrayer$^-$.FictionalCharacter
12	PrixArtiste	$\equiv$	Award
13	Artiste $\sqcap$ $\exists$aPourRecompensePersonne.PrixArtiste	$\equiv$	Artist $\sqcap$ $\exists$award.Award
14	FilmLocalise	$\sqsubseteq$	Film
15	FilmLocalise	$\sqsubseteq$	Film $\sqcap$ ($\exists$basedOn.Film $\sqcup$ $\exists$previousWork.Film)
16	$\forall$x FilmLocalise(x)	$\rightarrow$	$\exists$yz Film(x) $\sqcap$originalLanguage(x, y) $\sqcap$ language(x, z) $\sqcap$ y $\neq$ z
17	Artiste $\sqcap$ $\exists$estDoubleurDans.FilmLocalise		N/A
18	Artiste $\sqcap$ $\exists$estRecompensePour.Film	$\sqsubseteq$	Artist $\sqcap$ $\exists$award(Award $\sqcap$ $\exists$related.Film)
19	Artiste $\sqcap$ $\exists$estRecompensePour.Film	$\sqsubseteq$	Artist $\sqcap$ $\exists$nominee$^-$.Film
20	Artiste $\sqcap$ $\exists$aPourSexe.Sexe	$\equiv$	Artist $\sqcap$ $\exists$sex.<xsd:string>
21	Artiste $\sqcap$ $\exists$estMembreDeJuryPourCompetition.Competition	$\sqsubseteq$	Person $\sqcap$ $\exists$organisationMember$^-$(Organisation $\sqcap$ $\exists$event.FilmFestival)
22	Jury	$\sqsubseteq$	Organisation
23	Artiste $\sqcap$ $\exists$estMembreDuJury.Jury	$\sqsubseteq$	Person $\sqcap$ $\exists$organisationMember$^-$.Organisation
24	Lieu	$\equiv$	Place
25	Artiste $\sqcap$ $\exists$aPourLieuNaissance.Lieu	$\equiv$	Artist $\sqcap$ $\exists$birthPlace.Place
26	Artiste $\sqcap$ $\exists$aPourLieuMort.Lieu	$\equiv$	Artist $\sqcap$ $\exists$deathPlace.Place
27	ActeurDeComplement	$\sqsubseteq$	Artist
28	ActeurDeComplement $\sqcap$ $\exists$contribueAuFilm.Film		?
29	ActeurDeComplement $\sqcap$ $\exists$estLeDoubleurDeActeur.Artiste	$\sqsubseteq$	Artist $\sqcap$ $\exists$dubber.Artist
30	ActeurDeComplement $\sqcap$ $\exists$estLeDoubleurDeRole.Role	$\sqsubseteq$	Artist $\sqcap$ $\exists$dubber.FictionalCharacter
31	Competition	$\equiv$	FilmFestival
32	PrixDecerne	$\equiv$	Award
33	Competition $\sqcap$ $\exists$decerne.PrixDecerne	$\equiv$	FilmFestival $\sqcap$ $\exists$event$^-$.Award
34	Competition $\sqcap$ $\exists$seDerouleDans.Lieu	$\equiv$	FilmFestival $\sqcap$ $\exists$location.Place
35	Competition $\sqcap$ $\exists$aPourJury.Jury	$\equiv$	FilmFestival $\sqcap$ $\exists$committee.xsd:string
36	owl:Thing $\sqcap$ $\exists$estRecompenseA.Competition	$\equiv$	Person $\sqcap$ $\exists$award(Award $\sqcap$ $\exists$event.FilmFestival)
37	Jury $\sqcap$ $\exists$deliberePourCompetition.Competition	$\sqsubseteq$	Organisation $\sqcap$ $\exists$event.FilmFestival
38	LongMetrageNomine	$\sqsubseteq$	Film
39	LongMetrageNomine $\sqcap$ $\exists$concourePourCompetition.Competition	$\sqsubseteq$	Film $\sqcap$ $\exists$film$^-$.FilmFestival
40	Film $\sqcap$ $\exists$estLocalisePour.Lieu	$\equiv$	Film $\sqcap$ $\exists$location.Place
41	Film $\sqcap$ $\exists$estUneLocalisationDe.Film		?
42	Film $\sqcap$ $\exists$estRecompenseA.Competition		N/A
43	Film $\sqcap$ $\exists$contribueAuFilm.Personne	$\equiv$	Work $\sqcap$ $\exists$associate.Person
44	Film $\sqcap$ $\exists$aPourRole.Role	$\equiv$	Film $\sqcap$ $\exists$starring(Person $\sqcap$ $\exists$portrayer$^-$.FictionalCharacter)
45	Film $\sqcap$ $\exists$aPourTechnique.Technique		N/A
46	Jury $\sqcap$ $\exists$aPourMembreDeJury.Personne	$\sqsubseteq$	Organisation $\sqcap$ $\exists$organisationMember.Person
47	PrixDecerne $\sqcap$ $\exists$estDecernePar.Competition	$\equiv$	Award $\sqcap$ $\exists$event.FilmFestival
48	PrixDecerne $\sqcap$ $\exists$recompenseFilm.Oeuvre		N/A
49	PrixDecerne $\sqcap$ $\exists$recompensePersonne.Personne	$\equiv$	Award $\sqcap$ $\exists$award$^-$.Artist
50	Role $\sqcap$ $\exists$apparaitDans.Film	$\equiv$	FictionalCharacter $\sqcap$ $\exists$portrayer(Actor $\sqcap$ $\exists$starring.Film)
51	Role $\sqcap$ $\exists$estIncarnePar.Artiste	$\equiv$	FictionalCharacter $\sqcap$ $\exists$portrayer.Person
52	Role $\sqcap$ $\exists$roleDoublePar.Doubleur	$\equiv$	FictionalCharacter $\sqcap$ $\exists$dubber$^-$.Person

Table 5.7: Correspondances complexes réalisées à la main entre Cinéma IRIT et DBpedia

1	∀ x,z (track_number(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberOfBombs(y,z)))
2	∀ x (Organization(x)	↔	∃ y(keyPerson(x,y) ⊓ OrganisationMember(y)))
3	∀ x,z (track_nnumber(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberOfCrew(y,z)))
4	∀ x,z (track_number(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberOfLaunches(y,z)))
5	∀ x (PositionChange(x)	↔	∃ y(owningOrganisation(x,y) ⊓ PoliticalParty(y)))
6	∀ x (time#Year(x)	↔	∃ y(event(x,y) ⊓ YearInSpaceflight(y)))
7	∀ x,z (ibdbId(x,z)	↔	∃ y (imdb(x,y) ⊓ valid(y,z)))
8	∀ x,z (numberOfTracks(x,z)	↔	∃ y (track(x,y) ⊓ track_number(y,z)))
9	∀ x (time#DateTimeDescription(x)	↔	∃ y(keyPerson(x,y) ⊓ VicePrimeMinister(y)))
10	∀ x,z (catalogue_number(x,z)	↔	∃ y (crewMember(x,y) ⊓ numberOfVisitors(y,z)))
11	∀ x,z (locationIdentifier(x,z)	↔	∃ y (publishing_location(x,y) ⊓ identifier(y,z)))
12	∀ x,z (activity_end(x,z)	↔	∃ y (endPoint(x,y) ⊓ crownDependency(y,z)))
13	∀ x,z (catalogue_number(x,z)	↔	∃ y (crewMember(x,y) ⊓ number(y,z)))
14	∀ x,z (track_number(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberBuilt(y,z)))
15	∀ x,z (activity_end(x,z)	↔	∃ y (endPoint(x,y) ⊓ endangeredSince(y,z)))
16	∀ x (FileFormat(x)	↔	∃ y(event(x,y) ⊓ FilmFestival(y)))
17	∀ x (time#DateTimeInterval(x)	↔	∃ y(keyPerson(x,y) ⊓ VicePrimeMinister(y)))
18	∀ x,z (activity_end(x,z)	↔	∃ y (endPoint(x,y) ⊓ dateExtended(y,z)))
19	∀ x,z (track_number(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberOfVisitors(y,z)))
20	∀ x,z (imdbId(x,z)	↔	∃ y (imdb(x,y) ⊓ valid(y,z)))
21	∀ x (OnlineGamingAccount(x)	↔	∃ y(keyPerson(x,y) ⊓ GaelicGamesPlayer(y)))
22	∀ x (event.owl#Factor(x)	↔	∃ y(keyPerson(x,y) ⊓ Actor(y)))
23	∀ x,z (track_number(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberOfRockets(y,z)))
24	∀ x,z (ibdbId(x,z)	↔	∃ y (imdb(x,y) ⊓ identifier(y,z)))
25	∀ x (Execution(x)	↔	∃ y(event(x,y) ⊓ Election(y)))
26	∀ x,z (imdbId(x,z)	↔	∃ y (imdb(x,y) ⊓ identifier(y,z)))

Table 5.8: Alignement complexe obtenu avec l'outil ComplexMapping entre l'ontologie Music et l'ontologie de DBpedia. L'alignement LOD a été utilisé comme alignement initial.

1	∀ x,z (track_number(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberOfBombs(y,z)))
2	∀ x (RecordingSession(x)	↔	∃ y(company(x,y) ⊓ RecordLabel(y)))
3	∀ x,z (track_number(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberOfCrew(y,z)))
4	∀ x (PositionChange(x)	↔	∃ y(owningOrganisation(x,y) ⊓ PoliticalParty(y)))
5	∀ x,z (track_number(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberOfLaunches(y,z)))
6	∀ x,z (ibdbId(x,z)	↔	∃ y (imdb(x,y) ⊓ valid(y,z)))
7	∀ x (time#DateTimeDescription(x)	↔	∃ y(keyPerson(x,y) ⊓ VicePrimeMinister(y)))
8	∀ x,z (numberOfTracks(x,z)	↔	∃ y (track(x,y) ⊓ track_number(y,z)))
9	∀ x,z (catalogue_number(x,z)	↔	∃ y (crewMember(x,y) ⊓ numberOfVisitors(y,z)))
10	∀ x,z (locationIdentifier(x,z)	↔	∃ y (publishing_location(x,y) ⊓ identifier(y,z)))
11	∀ x,z (activity_end(x,z)	↔	∃ y (endPoint(x,y) ⊓ crownDependency(y,z)))
12	∀ x,z (catalogue_number(x,z)	↔	∃ y (crewMember(x,y) ⊓ number(y,z)))
13	∀ x,z (track_number(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberBuilt(y,z)))
14	∀ x,z (activity_end(x,z)	↔	∃ y (endPoint(x,y) ⊓ endangeredSince(y,z)))
15	∀ x (FileFormat(x)	↔	∃ y(event(x,y) ⊓ FilmFestival(y)))
16	∀ x (time#DateTimeInterval(x)	↔	∃ y(keyPerson(x,y) ⊓ VicePrimeMinister(y)))
17	∀ x,z (activity_end(x,z)	↔	∃ y (endPoint(x,y) ⊓ dateExtended(y,z)))
18	∀ x,z (track_number(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberOfVisitors(y,z)))
19	∀ x,z (imdbId(x,z)	↔	∃ y (imdb(x,y) ⊓ valid(y,z)))
20	∀ x (Record(x)	↔	∃ y(company(x,y) ⊓ RecordLabel(y)))
21	∀ x (OnlineGamingAccount(x)	↔	∃ y(keyPerson(x,y) ⊓ GaelicGamesPlayer(y)))
22	∀ x (event.owl#Factor(x)	↔	∃ y(keyPerson(x,y) ⊓ Actor(y)))
23	∀ x (MusicalExpression(x)	↔	∃ y(keyPerson(x,y) ⊓ MusicalArtist(y)))
24	∀ x,z (track_number(x,z)	↔	∃ y (aircraftBomber(x,y) ⊓ numberOfRockets(y,z)))
25	∀ x (Recording(x)	↔	∃ y(company(x,y) ⊓ RecordLabel(y)))
26	∀ x,z (ibdbId(x,z)	↔	∃ y (imdb(x,y) ⊓ identifier(y,z)))
27	∀ x (Execution(x)	↔	∃ y(event(x,y) ⊓ Election(y)))
28	∀ x,z (imdbId(x,z)	↔	∃ y (imdb(x,y) ⊓ identifier(y,z)))

Table 5.9: Alignement complexe obtenu avec l'outil ComplexMapping entre l'ontologie Music et l'ontologie de DBpedia. Nous avons utilisé la fusion des alignements générés par les outils de l'OAEI comme alignement initial.

Chapter 6

Conclusions et perspectives

Dans ce mémoire, nous avons présenté une approche de réécriture automatique de patrons de requêtes à l'aide d'alignements d'ontologies. Dans un premier temps, nous avons utilisé les alignements issus d'un ensemble de systèmes d'alignement capables de générer des correspondances simples. Nous avons pu observer que ce type de correspondance est insuffisant pour notre tâche de réécriture.

Dans un deuxième temps, nous avons utilisé une approche proposée dans la littérature qui est basée sur l'utilisation des patrons de correspondances complexes, afin de générer des correspondances plus expressives et mieux adaptées à notre tâche. Pour pouvoir évaluer cette approche, nous avons construit manuellement un ensemble de correspondances complexes entre les ontologies Cinéma IRIT et Music d'une part, et DBpedia d'autre part. Ces alignements constituent des ressources pouvant se révéler utiles dans le cadre d'évaluations de systèmes d'alignements générant ce type de correspondances. Cependant, l'approche a généré de très mauvais résultats pour les ontologies considérées, ce qui ouvre plusieurs pistes pour les travaux futurs, notamment :

- Proposer une extension des approches basées sur les patrons de correspondances complexes en proposant la définition de patrons de correspondances qui considèrent les extensions des ontologies (les instances) ;
- Utiliser les systèmes d'alignement capables de générer des correspondances entre les instances et combiner leurs résultats avec les patrons de correspondances complexes qui traitent les niveaux terminologique et structurel des ontologies;
- Proposer un mécanisme pour compléter les alignements non couvrants générés pour les ontologies en question à partir de l'exploitation de l'espace des patrons possibles contenant les concepts dans les patrons incomplets;
- Développer un système d'alignement capable de générer des correspondances complexes en utilisant les nouvelles approches proposées;
- Considérer l'intervention de l'utilisateur lors du processus de complétion de patrons et permettre l'apprentissage automatique de nouveaux patrons.

Concernant le système SWIP, une évolution possible à moindre coût pourrait être le support de la disjonction dans les sous-patrons de requêtes. La disjonction est cependant possible en l'état par l'ajout

d'un sous-patron optionnel pour chacun des opérandes de la disjonction.
Nous n'avons pas réellement pu évaluer nos patrons pour DBpedia dans SWIP, du fait de l'étape préliminaire d'annotation sémantique qui échoue à trouver les entités et les individus dans DBpedia correspondant aux mots-clefs de la requête. L'intégration dans SWIP d'un outil d'annotation sémantique comme DBpedia Spotlight[1] permettrait de résoudre ce problème.

Remerciements

Je remercie Cassia Trojahn, Ollivier Haemmerlé, et Camille Pradel pour leur accueil, les conseils et les remarques toujours pertinentes qu'ils m'ont prodigués, et la disponibilité dont ils ont fait preuve tout au long de mon stage, et ce malgré un emploi du temps très chargé. Ils m'ont permis de mettre un pied dans le monde de la recherche, dans un domaine qui m'intéresse tout particulièrement, et de mener à bien mon projet de fin d'études dans les meilleures conditions.

Je remercie également tous les membres de l'équipe MELODI[2] pour leur accueil et la découverte de leurs travaux de recherche.

[1] http://fr.wikipedia.org/wiki/DBpedia_Spotlight
[2] http://www.irit.fr/-Equipe-MELODI-

Bibliography

[Aguirre et al., 2012] Aguirre, J. L., Eckert, K., Euzenat, J., Ferrara, A., van Hage, W. R., Hollink, L., Meilicke, C., Nikolov, A., Ritze, D., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Sváb-Zamazal, O., Trojahn, C., Jiménez-Ruiz, E., Grau, B. C., and Zapilko, B. (2012). Results of the ontology alignment evaluation initiative 2012. In *Proceedings of the 7th International Workshop on Ontology Matching*.

[Athanasis et al., 2004] Athanasis, N., Christophides, V., and Kotzinos, D. (2004). Generating on the fly queries for the semantic web: The ics-forth graphical rql interface (grql). In *The Semantic Web–ISWC 2004*, pages 486–501. Springer.

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, New York, NY, USA.

[Cabrio et al., 2012] Cabrio, E., Cojan, J., Aprosio, A. P., Magnini, B., Lavelli, A., and Gandon, F. (2012). Qakis: an open domain qa system based on relational patterns.

[Clemmer and Davies, 2011] Clemmer, A. and Davies, S. (2011). Smeagol: A "specific-to-general" semantic web query interface paradigm for novices. In *Database and Expert Systems Applications*, pages 288–302. Springer.

[Correndo and Shadbolt, 2011] Correndo, G. and Shadbolt, N. (2011). Translating expressive ontology mappings into rewriting rules to implement query rewriting. In *Proceedings of the 6th International Workshop on Ontology Matching, Bonn, Germany, October 24*.

[David et al., 2011] David, J., Euzenat, J., Scharffe, F., and Trojahn, C. (2011). The alignment api 4.0. *Semantic Web*, 2(1):3–10.

[Dhamankar et al., 2004] Dhamankar, R., Lee, Y., Doan, A., Halevy, A., and 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, pages 383–394, New York, NY, USA. ACM.

[Dong et al., 2005] Dong, X., Halevy, A., and Madhavan, J. (2005). Reference reconciliation in complex information spaces. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 85–96, New York, NY, USA. ACM.

[Elbassuoni and Blanco, 2011] Elbassuoni, S. and Blanco, R. (2011). Keyword search over rdf graphs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 237–242. ACM.

[Elbassuoni et al., 2010] Elbassuoni, S., Ramanath, M., Schenkel, R., and Weikum, G. (2010). Searching rdf graphs with sparql and keywords. *IEEE Data Eng. Bull.*, 33(1):16–24.

[Euzenat et al., 2011] Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., and dos Santos, C. T. (2011). Ontology alignment evaluation initiative: Six years of experience. *J. Data Semantics*, 15:158–192.

[Euzenat and Shvaiko, 2007] Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer-Verlag, Berlin, Heidelberg.

[Ferré and Hermann, 2011] Ferré, S. and Hermann, A. (2011). Semantic search: reconciling expressive querying and exploratory search. In *The Semantic Web–ISWC 2011*, pages 177–192. Springer.

[Ferré and Hermann, 2012] Ferré, S. and Hermann, A. (2012). Reconciling faceted search and query languages for the semantic web. *International Journal of Metadata, Semantics and Ontologies*, 7(1):37–54.

[Ferré et al., 2011] Ferré, S., Hermann, A., and Ducassé, M. (2011). Combining faceted search and query languages for the semantic web. In *Advanced Information Systems Engineering Workshops*, pages 554–563. Springer.

[Kalfoglou and Schorlemmer, 2003] Kalfoglou, Y. and Schorlemmer, M. (2003). Ontology mapping: the state of the art. *Knowl. Eng. Rev.*, 18(1):1–31.

[Lehmann and Bühmann, 2011] Lehmann, J. and Bühmann, L. (2011). Autosparql: Let users query your knowledge base. In *The Semantic Web: Research and Applications*, pages 63–79. Springer.

[Lei et al., 2006] Lei, Y., Uren, V., and Motta, E. (2006). Semsearch: A search engine for the semantic web. In *Managing Knowledge in a World of Networks*, pages 238–245. Springer.

[Pradel et al., 2012a] Pradel, C., Haemmerlé, O., Hernandez, N., et al. (2012a). Des patrons modulaires de requêtes sparql dans le système swip. *23es Journées Francophones d'Ingénierie des Connaissances*.

[Pradel et al., 2011] Pradel, C., Haemmerlé, O., and Hernandez, N. (2011). Expressing conceptual graph queries from patterns: How to take into account the relations. In *Conceptual Structures for Discovering Knowledge*, Lecture Notes in Computer Science, pages 229–242. Springer Berlin Heidelberg.

[Pradel et al., 2012b] Pradel, C., Haemmerlé, O., and Hernandez, N. (2012b). A semantic web interface using patterns: The swip system. In *Graph Structures for Knowledge Representation and Reasoning*, Lecture Notes in Computer Science, pages 172–187. Springer Berlin Heidelberg.

[Pradel et al., 2012c] Pradel, C., Hernandez, N., Kamel, M., and Rothenburger, B. (2012c). Une ontologie du cinéma pour évaluer les applications du web sémantique. In *Atelier Ontologies et Jeux de Données pour évaluer le web sémantique, IC*.

[Qin et al., 2007] Qin, H., Dou, D., and LePendu, P. (2007). Discovering executable semantic mappings between ontologies. In *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I*, OTM'07, pages 832–849, Berlin, Heidelberg. Springer-Verlag.

[Rahm and Bernstein, 2001] Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350.

[Ritze et al., 2009] Ritze, D., Meilicke, C., Sváb-Zamazal, O., and Stuckenschmidt, H. (2009). A pattern-based ontology matching approach for detecting complex correspondences. In *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) collocated with the 8th International Semantic Web Conference (ISWC-2009) Chantilly, USA, October 25, 2009*.

[Ritze et al., 2010] Ritze, D., Völker, J., Meilicke, C., and Sváb-Zamazal, O. (2010). Linguistic analysis for complex ontology matching. In *Proceedings of the 5th International Workshop on Ontology Matching (OM-2010), Shanghai, China, November 7, 2010*.

[Russell and Smart, 2008] Russell, A. and Smart, P. R. (2008). Nitelight: A graphical editor for sparql queries. In *Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC2008), Karlsruhe, Germany, October 28, 2008*.

[Scharffe, 2009] Scharffe, F. (2009). *Correspondence Patterns Representation*. PhD thesis, University of Innsbruck, Innsbruck.

[Tran et al., 2009] Tran, T., Wang, H., Rudolph, S., and Cimiano, P. (2009). Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 405–416. IEEE.

[Walshe, 2012] Walshe, B. (2012). Identifying complex semantic matches. In *Proceedings of the 9th international conference on The Semantic Web: research and applications*, pages 849–853. Springer-Verlag.

[Wang et al., 2008] Wang, H., Zhang, K., Liu, Q., Tran, T., and Yu, Y. (2008). Q2semantic: A lightweight keyword interface to semantic search. In *The Semantic Web: Research and Applications*, pages 584–598. Springer.

[Zhou et al., 2007] Zhou, Q., Wang, C., Xiong, M., Wang, H., and Yu, Y. (2007). Spark: adapting keyword query to semantic search. In *The Semantic Web*, pages 694–707. Springer.