UNIVERSITÉ DE MONTRÉAL

EXTRACTION D'AXIOMES ET DE RÈGLES LOGIQUES À PARTIR DE DÉFINITIONS DE WIKIPÉDIA EN LANGAGE NATUREL

LARA HAIDAR-AHMAD

DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APLIQUÉES

(GÉNIE INFORMATIQUE)

AVRIL 2017

© Lara Haidar-Ahmad, 2017.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

EXTRACTION D'AXIOMES ET DE RÉGLES LOGIQUES À PARTIR DE DÉFINITIONS DE WIKIPÉDIA EN LANGAGE NATUREL

présenté par : HAIDAR-AHMAD Lara

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. DESMARAIS Michel C., Ph. D, président

M. GAGNON Michel, Ph. D., membre et directeur de recherche

Mme ZOUAQ Amal, Ph. D., membre et codirectrice de recherche

M. LANGLAIS Philippe, Ph. D., membre

REMERCIEMENTS

Je tiens à remercier mes directeurs de recherche Michel Gagnon et Amal Zouaq pour leur encadrement et soutien tout au long de mon travail. Je remercie mes coéquipiers de laboratoire pour leurs aides à évaluer ce travail : Mohamed Chabchoub, Anthony Garant, Konstantinos Lambrou-Latreille, Simon Malenfant-Corriveau, Erwan Marchand, et finalement Ludovic Font, avec qui j'ai collaboré durant ma maîtrise pour l'écriture d'un des articles présentés dans ce mémoire. Finalement, je tiens à remercier mes parents et mes sœurs qui m'ont soutenu et m'ont encouragé tout au long de mes études.

RÉSUMÉ

Le Web sémantique repose sur la création de bases de connaissances complexes reliant les données du Web. Notamment, la base de connaissance DBpedia a été créée et est considérée aujourd'hui comme le « noyau du réseau Linked Open Data ». Cependant DBpedia repose sur une ontologie très peu riche en définitions de concepts et ne prend pas en compte l'information textuelle de Wikipedia. L'ontologie de DBpedia contient principalement des liens taxonomiques et des informations sur les instances. L'objectif de notre recherche est d'interpréter le texte en langue naturelle de Wikipédia, afin d'enrichir DBpedia avec des définitions de classes, une hiérarchie de classes (relations taxonomiques) plus riche et de nouvelles informations sur les instances. Pour ce faire, nous avons recours à une approche basée sur des patrons syntaxiques implémentés sous forme de requêtes SPARQL. Ces patrons sont exécutés sur des graphes RDF représentant l'analyse syntaxique des définitions textuelles extraites de Wikipédia. Ce travail a résulté en la création de AXIOpedia, une base de connaissances expressive contenant des axiomes complexes définissant les classes, et des triplets *rdf:type* reliant les instances à leurs classes.

ABSTRACT

The Semantic Web relies on the creation of rich knowledge bases which links data on the Web. In that matter, DBpedia started as a community effort and is considered today as the central interlinking hub for the emerging Web of data. However, DBpedia relies on a lighweight ontology and deals with some substantial limitations and lacks some important information that could be found in the text and the unstructured data of Wikipedia. Furthermore, the DBpedia ontology contains mainly taxonomical links and data about the instances, and lacks class definitions. The objective of this work is to enrich DBpedia with class definitions and taxonomical links using text in natural language. For this purpose, we rely on a pattern-based approach that transforms textual definitions from Wikipedia into RDF graphs, which are processed to query syntactical pattern occurrences using SPARQL. This work resulted in the creation of AXIOpedia, a rich knowledge base containing complex axioms defining classes and *rdf:type* relations relating instances with these classes.

TABLE DES MATIÈRES

REMERCIEMENTS III
RÉSUMÉ IV
ABSTRACT V
TABLE DES MATIÈRES VI
LISTE DES TABLEAUX X
LISTE DES FIGURES XII
LISTE DES SIGLES ET ABRÉVIATIONS XIII
CHAPITRE 1 INTRODUCTION 1
1.1 Problématique et objectif de recherche 2
1.2 Plan du mémoire 3
CHAPITRE 2 REVUE DE LITTÉRATURE 5
2.1 Les technologies du Web sémantique 5
2.2 Importance des techniques automatiques d'extraction 7
2.3 Travaux reliés à notre recherche 10
2.3.1 Distinction entre classes et instances 10
2.3.2 Extraction d'axiomes définissant des classes 10
2.3.3 Extraction de types d'instances 11
2.3.4 Désambiguïsation 11
CHAPITRE 3 MÉTHODOLOGIE 14
3.1 Patrons syntaxiques 14
3.2 SPARQL et graphe RDF 14
3.3 Architecture générale 15
3.4 Base de connaissances AXIOpedia 18

3.5 Service Web 18

3.6 Présentation des articles 18

3.7 Évaluation 19

CHAPITRE 4 ARTICLE 1: AUTOMATIC EXTRACTION OF AXIOMS FROM WIKIPEDIA USING SPARQL 21

4.1 Introduction 21

4.2 Methodology 22

4.3 Preliminary Evaluation and Discussion 24

4.4 Conclusion and Future Work 25

4.5 Acknowledgement 25

4.6 References 25

CHAPITRE 5 ARTICLE 2: ENTITY TYPING AND LINKING USING SPARQL PATTERNS AND DBPEDIA 27

5.1 Introduction 27

5.2 Related Work 28

5.2.1 Type extraction 28

5.2.2 Type alignment 29

5.3 Entity Type Extraction 29

5.3.1 Sentence Graph Representation 29

5.3.2 Pattern Identification 30

5.4 Type alignment 32

5.4.1 DBpedia Type Identification 33

5.4.2 Dolce+DUL Alignment 35

5.5 Evaluation 36

5.5.1 Type extraction evaluation 36

5.5.2 Type alignment evaluation 38

5.5.3 Overall results 39

5.6 Discussion 42

5.7 Conclusion 44

5.8 Acknowledgement 45

5.9 References 45

CHAPITRE 6 ARTICLE 3: AXIOPEDIA: ENRICHING DBPEDIA WITH OWL AXIOMS FROM WIKIPEDIA 47

6.1 Introduction 47

6.2 Related Work 49

6.3 Overview and Motivation of this Work 52

6.4 Methodology and General Architecture 54

6.4.1 Identification of Instances and Classes in Wikipedia 54

6.4.2 Processing the Textual Definitions 57

6.4.3 Processing the RDF Graph to Extract Axioms for Classes 58

6.4.4 Processing the RDF Graph to Extract rdf:type Triples for Instances 65

6.4.5 Hearst Patterns 67

6.4.6 Axiom and Type Disambiguation 67

6.5 Evaluation and Results 68

6.5.1 Evaluation of the process of distinguishing Instances and Classes 69

6.5.2 Evaluation of the Axiom Extraction 70

6.5.3 Type Extraction Evaluation 75

6.5.4 Disambiguation Evaluation 78

6.5.5 Inter-Annotator Agreement 78

6.6 Conclusion and Future Work 81

6.7 References 82

6.8 Annex 86

CHAPITRE 7 DISCUSSION GÉNÉRALE 97

7.1 Analyse et discussion des résultats.....97

7.2 Retour sur les objectifs et questions de recherche.....97

7.3 Travail futur et améliorations proposées 98

CHAPITRE 8 CONCLUSION 100

BIBLIOGRAPHIE 101

LISTE DES TABLEAUX

Table 4.1 Most Frequent Patterns and their Respective Axioms 23

Table 4.2 Evaluation Results 24

Table 4.3 Axioms' Precision based on Levenstein Similarity with the Gold Standard 24

Table 5.1 Most Frequent Patterns Describing an Entity/Type Relationship 31

Table 5.2 Statistics for the type extraction evaluation on the train dataset 37

Table 5.3 Statistics for the type extraction evaluation on the evaluation dataset 38

Table 5.4 Comparison of each method for type alignment on the OKE challenge train dataset ... 38

Table 5.5 Overall precision, recall and F-measure computed using Gerbil on the 2016 train dataset 39

Table 5.6 Overall precision, recall and F-measure computed using Gerbil on the 2016 test dataset 39

Table 5.7 Overall precision, recall and F-measure for the two participating systems and the baseline CETUS on the test dataset 40

Table 5.8 Overall precision, recall and F-measure for the two participating systems and the baseline CETUS on the corrected test dataset 41

Table 6.1 Excerpt of the Patterns and their Mapped Axioms. The Complete list is presented in Table 6.13 (Annex I). 61

Table 6.2 Patterns for detecting Instances' Types 66

Table 6.3 Results for the Wikipedia Pages Filtering 69

Table 6.4 Result for Manual Evaluation of the Built Axioms 71

Table 6.5 Result for the Types' Extraction for the Evaluation based on DBpedia 75

Table 6.6 Result for the OKE 2016 Evaluation 76

Table 6.7 Result for the Types' Extraction Manual Evaluation 77

Table 6.8 Result for the Disambiguation Manual Evaluation 78

Table 6.9 Agreement for the Page Filtering Evaluation 79

Table 6.10 Agreement for the Axioms' Evaluation 79

Table 6.11 Agreement for the Type Extraction Manual Evaluation 80

Table 6.12 Agreement for the Disambiguation Manual Evaluation 80

Table 6.13 Patterns and their mapped axioms 86

Table 6.14 Patterns for detecting Instances' Types 92

Table 6.15 Hearst Patterns and their mapped axioms 93

LISTE DES FIGURES

Figure 2.1 La pile du Web sémantique 6

Figure 2.2 Ontology Layer Cake 8

Figure 2.3 Pipeline des méthodes de désambiguïsation 12

Figure 3.1 Architecture générale 17

Figure 4.1 The RDF Representation of the definition of vehicles. 22

Figure 4.2 Pipeline 22

Figure 5.1 The RDF Representation of the definition of Skara Cathedral 30

Figure 6.1 Ontology Layer Cake 50

Figure 6.2 General Architecture 56

Figure 6.3 RDF Graph of the definition of Natural Satellite 58

Figure 6.4 SPARQL implementation of Pattern (4) 63

LISTE DES SIGLES ET ABRÉVIATIONS

LOD	Linked Open Data
OKE	Open Knowledge Extraction
OWL	Web Ontology Language
POS	Part-of-Speech
RDF	Resource Description Framework
RDFS	RDF Schema
SPARQL	SPARQL Protocol and RDF Query Language

CHAPITRE 1 INTRODUCTION

Le Web sémantique est un champ de recherche relativement récent qui vise à développer un Web dans lequel les informations sont structurées, de façon à permettre aux machines de les traiter de manière automatique et d'effectuer des inférences à partir de ces données. C'est un Web de données qui se base sur des concepts plutôt que sur les mots-clefs. Par exemple des concepts comme *Femme*, *Mère* et *Enfant* seraient représentés et reliés par des relations comme *Mère* es*t une Femme* et *a au moins un enfant*. Le terme "Web sémantique" a été inventé par Tim Berners-Lee, un des principaux inventeurs du World Wide Web. Ce dernier explique que :

> Le Web Sémantique n'est pas seulement l'introduction de données dans le Web. Il s'agit plutôt de faire des liens […]. Avec le Linked Data, il suffirait d'avoir quelques données pour en déduire d'autres en rapport avec elles[1]. (Tim Berners-Lee, 2006)

En effet, les données étant structurées et interreliées, cela permet d'inférer de nouvelles informations à partir des données déjà présentes sur le Web, et ainsi de repousser les limites du Web d'aujourd'hui. En reprenant l'exemple précédant, si nous avons pour informations qu'une instance *Jocaste* est une *Femme* et que l'instance *Jocaste* a un *Enfant Œdipe*, on pourrait en inférer que *Jocaste* est une *Mère*. Ainsi, un des plus importants défis du Web sémantique est l'enrichissement et la croissance du *Linked Open Data* (LOD). Le LOD est une initiative qui vise à relier les données du Web sémantique entre elles, et représente ainsi la concrétisation du Web sémantique dans sa forme actuelle. En effet, il n'est non seulement nécessaire d'extraire les données du Web, mais il est aussi essentiel de faire des liens entre ces données. Nous voyons donc la nécessité de construire des bases de connaissances assez riches permettant d'effectuer de meilleures recherches et inférences. Afin d'être en mesure de construire de telles bases de connaissances, il faudrait tout d'abord extraire les données déjà présentes sur le Web et les intégrer aux bases de connaissances. Wikipédia étant l'encyclopédie du Web parmi les plus reconnues, elle est une source parfaite pour cette tâche. En effet, Wikipédia offre des articles qui en majorité sont

---

[1] Traduit de https://www.w3.org/DesignIssues/LinkedData.html [2006-07-27]

constitués des textes en langue naturelle, mais qui contiennent aussi différents types de données structurées, comme des boîtes d'information (infoboxes), des catégories, des images, et des liens vers des pages externes.

C'est dans ce contexte que DBpedia a vu le jour. DBpedia, qui est aujourd'hui considérée comme le « noyau du LOD », est un projet né d'un effort collectif qui vise à extraire les informations de Wikipédia, et à les représenter par des triplets annotés en RDF (Resource Description Framework), un modèle de données du Web sémantique que nous présenterons plus tard. DBpedia se structure autour d'une ontologie qui vise à décrire les classes les plus importantes (comme les classes *dbo:Person*, *dbo:Country*, *dbo:City*). Cependant, cette ontologie est loin d'être exhaustive et représente essentiellement une taxonomie de classes nommées ainsi que des propriétés d'objets, tout en manquant d'informations essentielles telles que les définitions de classes. En effet, les instances de DBpedia sont généralement typées et reliées à leurs classes respectives par des triplets RDF, sans que ces classes ne soient définies. DBpedia gagnerait en qualité si on ajoutait des axiomes logiques qui serviraient à documenter les classes, et qui aideraient à inférer de nouvelles informations concernant les instances de ces classes. Par ailleurs, jusqu'à 20% des instances de DBpedia ne sont pas typées [1], et gagneraient à être reliées à des classes par des liens de typage.

1.1 Problématique et objectif de recherche

Une des motivations de notre travail est d'enrichir DBpedia en y incluant des axiomes pour définir les classes et des liens taxonomiques pour typer les instances à partir des définitions de Wikipédia. Afin d'atteindre cet objectif, nous tentons de répondre aux questions de recherche suivantes :

- La structure grammaticale d'une phrase est-elle suffisante pour extraire des axiomes et des liens taxonomiques en y identifiant des patrons syntaxiques récurrents ?

- Quels sont les patrons syntaxiques les plus récurrents dans les définitions extraites de Wikipédia ?

- Est-il avantageux de profiter de la simplicité de RDF et SPARQL pour la recherche d'occurrences de patrons syntaxiques dans une phrase ?

Dans ce contexte, notre travail vise à extraire des axiomes à partir des définitions de Wikipédia (la première phrase du premier paragraphe), en se basant sur les liens grammaticaux entre les mots de ces définitions. Nous extrayons ainsi des axiomes et des liens taxonomiques afin de créer une base de connaissance, AXIOpedia, qui serait intégrable à DBpedia et qui permettrait d'enrichir DBpedia et son ontologie.

Nous présentons une approche basée sur des patrons syntaxiques : nous représentons les définitions sous forme de graphes RDF mettant en valeurs les relations grammaticales entre les mots de la phrase et nous identifions des patrons grammaticaux et syntaxiques que nous cherchons dans ce graphe. Chaque patron est associé à un axiome ou un ensemble d'axiomes que nous construisons lors de la détection de ce patron. Pour la recherche des occurrences des patrons dans le graphe RDF, nous représentons les patrons sous forme de requêtes SPARQL (SPARQL Protocol and RDF Query Language), un langage de requête qui permet de manipuler des données RDF.

Un autre objectif de notre travail est d'offrir un service Web[2] permettant aux chercheurs du domaine d'extraire automatiquement à partir de pages Wikipédia, des axiomes représentés dans le langage de logique descriptive OWL présenté plus tard.

1.2 Plan du mémoire

Ce mémoire est constitué de huit chapitres présentant les étapes de notre travail. Nous discutons tout d'abord, dans le second chapitre, de la revue de littérature ; nous y présentons les technologies du Web sémantique, ainsi que les travaux existants pertinents pour notre domaine de recherche. Par la suite, dans le troisième chapitre, nous expliquons la méthodologie et les étapes de recherche, tout en faisant référence aux trois articles qui détaillent notre travail, présentés dans les trois chapitres qui suivent. Le quatrième chapitre présente l'article *Automatic Extraction of Axioms from Wikipedia Using SPARQL*, publié dans le cadre de la conférence *Extended Semantic Web Conference (ESWC)* en 2016. Cet article constitue une première version de notre travail de recherche sur l'extraction d'axiomes à partir de textes. Le cinquième chapitre contient l'article

[2] www.westlab.polymtl.ca/AXIOpediaWebApp/Home

Entity Typing and Linking Using SPARQL Patterns and DBpedia, publié dans le cadre de la compétition *Open Knowledge Extraction* (*OKE*) de la conférence *Extended Semantic Web Conference (ESWC)* en 2016. Cet article se concentre sur le typage des instances, et présente notre approche, qui a obtenu les meilleures performances de la compétition. Le sixième chapitre présente l'article *AXIOpedia: Enriching DBpedia with OWL Axioms from Wikipedia*, soumis au *Semantic Web Journal*, qui est la synthèse de notre travail, et qui propose un approfondissement de l'extraction d'axiomes complexes. Dans le septième chapitre, nous discutons des résultats obtenus, des travaux futurs, ainsi que des questions de recherche, avant de conclure sur ce travail dans le chapitre 8.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre introduit les travaux existants en lien avec notre étude. Dans un premier temps, nous présentons les technologies du Web sémantique sur lesquelles notre travail se base. La seconde section aborde les types de travaux sur l'extraction d'information et leur importance. La troisième section présente les différents travaux en liens avec notre recherche.

2.1 Les technologies du Web sémantique

Afin de rendre le Web sémantique possible, le consortium W3C a défini plusieurs langages permettant de représenter la sémantique des données qui s'y trouvent. La *Semantic Web stack* [2] ou pile du Web sémantique (Figure 2.1), expose les principales composantes du Web sémantique et illustre l'architecture des langages qui ont été conçus pour sa réalisation. Dans cette architecture, chaque couche exploite et utilise la couche précédente. Parmi les couches, on compte, RDF, RDFS, OWL et SPARQL.

RDF [3] est un langage permettant de présenter un modèle de données sous forme de triplets reliant des entités entre elles à partir de propriétés. Un ensemble de triplets RDF mettant en relation des ressources est ce qu'on appelle un graphe RDF. Grâce à RDF, nous pouvons exprimer des connaissances et des relations comme par exemple *:Ottawa :estCapitalDe :Canada*. RDFS (RDF Schema) [4], étend RDF pour permettre de créer des hiérarchies de classes ou de propriétés ou de définir le domaine et l'image des propriétés ; comme par exemple *:estCapitalDe rdfs:Range :Pays* ou encore :*Pays rdfs:SubclassOf :Lieu*.

OWL (Web Ontology Language) [5] est le langage à la tête de la hiérarchie des langages, puisqu'il étend RDFS et augmente son expressivité en ajoutant des contraintes pour décrire des classes, des propriétés, et des triplets RDF, comme par exemple ajouter des cardinalités, des égalités entre des classes ou propriétés, etc. L'expressivité de OWL permet donc de représenter des relations telles que *[rdf:type owl:Restriction ; owl2:onClass :Parent ; owl:onProperty :aEnfant ; owl:minCardinalityQ "1"]*, ou *:Parent owl:equivalentClass (:Mère ∪ :Père)* (où le symbole ∪ représente une union). OWL est un langage basé sur une logique descriptive, permettant de construire des bases de connaissance et des ontologies riches et complexes. La logique

descriptive est un formalisme pour représenter des connaissances de manière formelle et structurée pour un domaine d'application, et est utilisée pour encoder une ontologie. Une ontologie est un ensemble de données représentant un domaine, et contenant deux types d'informations : les axiomes de la TBox (terminological box), qui définissent des concepts en spécifiant les propriétés qui les caractérisent (par exemple : une *mère* est une *femme* qui a *au moins* un *enfant*) et les axiomes de la ABox (assertion box), qui fournissent des informations sur les instances (par exemple : *Marie est une femme* et *Paul est un enfant de Marie*). Le Web contient plusieurs ontologies OWL de tailles différentes, contenant des données de la ABox et TBox, et qui sont parfois reliées entre elles et se font référence.

SPARQL [6] est un langage de requête permettant de manipuler des données RDF. En particulier, il permet de rechercher des triplets dans un graphe RDF, d'en créer, et d'en supprimer. SPARQL est utilisé pour accéder aux bases de connaissances.

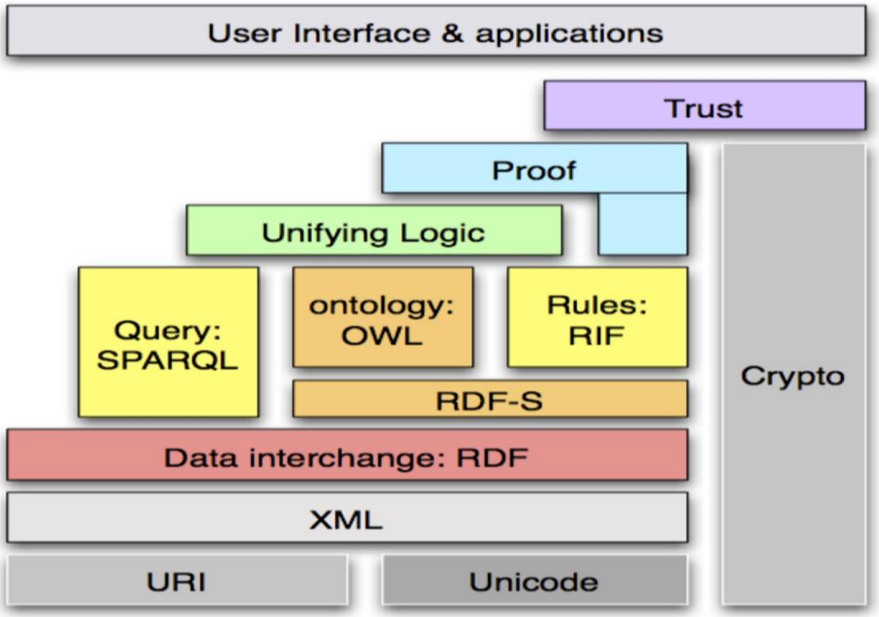

Figure 2.1 La pile du Web sémantique[3]

[3] https://www.w3.org/2009/Talks/0120-campus-party-tbl/#(14)

2.2 Importance des techniques automatiques d'extraction

Wikipédia étant la plus large encyclopédie collaborative du Web, de nombreux travaux de recherche se concentrent sur l'extraction d'information à partir de celle-ci, pour enrichir les bases de connaissances telles que DBpedia, YAGO [7], et autres. L'article [8] explique l'importance de Wikipédia dans ce domaine en montrant ce qui, parmi ses caractéristiques et sa structure, favorise l'extraction d'information. En effet, Wikipédia est une des plus larges encyclopédies du Web, elle est constamment mise à jour par une communauté d'internautes et est constituée en majorité de texte. Mais elle contient aussi un grand nombre de données structurées telles que le système de catégories, les boîtes d'information (infoboxes) contenants des données sous forme de clef-valeur, et les hyperliens, qui permettent de structurer le contenu de Wikipédia et ainsi de faciliter l'extraction d'information. On constate donc que Wikipédia contient un grand nombre d'informations, ce qui explique l'intérêt porté par les chercheurs qui visent à y extraire les connaissances; on compte un grand nombre de recherches, que ce soit des travaux qui se concentrent sur l'extraction d'informations à partir des données structurées telles que les boîtes d'information (infoboxes) et les systèmes de catégories [12], ou plutôt des travaux qui se focalisent sur l'extraction d'information à partir du texte en ayant recours à une analyse sémantique, la recherche de patrons syntaxiques, et des techniques d'apprentissage machine [13] [14].

Un de ces travaux a abouti à la création de DBpedia [12], la représentation RDF de Wikipédia. [9] évalue la qualité de DBpedia et estime dans une de leurs évaluations que les ressources de DBpedia possèdent en moyenne 47.19 triplets, un chiffre largement plus haut que la majorité des autres ensembles de connaissance existants. Cette étude explique aussi que DBpedia contient de nombreuses erreurs et inconsistances dans les données. Les auteurs estiment que la description d'une entité dans DBpedia possède en moyenne 5.69 erreurs. De plus, DBpedia contient essentiellement des données extraites des informations structurées de Wikipédia [10], alors que des informations importantes peuvent être extraites du texte. Enfin, DBpedia contient principalement des données sur les instances et ne possède pas assez de définitions de classes et axiomes. Cependant les données sur les instances ne suffisent pas dans une base de connaissances riche qui offrirait les mécanismes d'inférence nécessaires au Web sémantique.

En effet, la construction d'une ontologie riche est une tâche complexe puisqu'il faut plusieurs types d'informations dans une ontologie ; Le *ontology layer cake* [11], présenté dans la figure 1, présente

les 6 couches de données essentielles dans une ontologie riche. Les six couches se regroupent pour former l'ensemble de connaissances requises dans une ontologie; la première couche contient des termes d'un domaine, auxquels les termes synonymes sont liés dans la seconde couche. Les termes font référence aux concepts qu'ils représentent dans la troisième couche. La quatrième couche permet d'organiser les concepts dans une hiérarchie. La cinquième couche connecte les instances entre elles par des relations. Finalement la dernière couche définit des axiomes et règles logiques.

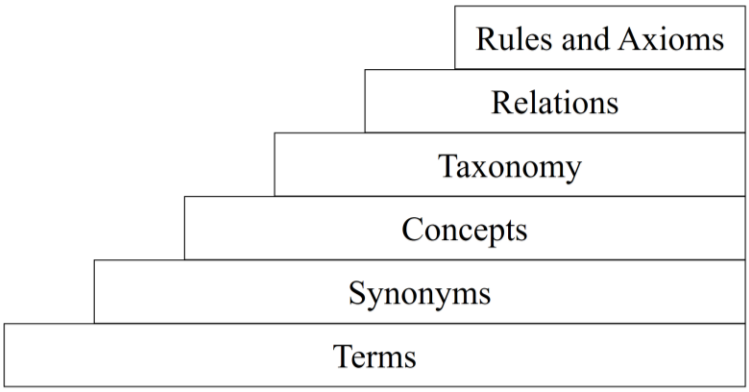

Figure 2.2 Ontology Layer Cake

Nous expliquons ci-dessous chacune des couches du *ontology layer cake*, ainsi que des exemples de travaux de recherches pour chacune. Une explication plus détaillée de ces couches se trouve dans l'article 3 (chapitre 6).

- **Termes:** Cette couche regroupe les unités lexicales représentant les concepts d'un domaine spécifique. Les travaux dans ce domaine se basent essentiellement sur des méthodes basées sur le traitement du langage et sur des méthodes statistiques [15] [16].
- **Synonymes:** Les termes faisant référence aux mêmes concepts sont définis comme étant des synonymes et sont détectés dans cette couche et reliés; on compte les travaux qui s'intéressent aux termes synonymes dans plusieurs langues [17] [18], ou encore la désambiguïsation lexicale de termes dans une même langue [19] [20].
- **Concepts:** Les travaux dans cette couche s'alignent avec les travaux des deux couches précédentes. Par concept on veut dire une classe d'instances avec certaines propriétés. Certains systèmes, comme KnowItAll [21], étendent les concepts existant en y ajoutant des instances de ce concept, d'autres extraient simultanément les instances et concepts [22].

- **Taxonomie:** Cette couche souligne l'importance de la création d'une hiérarchie qui structure les concepts et relie les instances à leurs classes. L'extraction de taxonomies repose essentiellement sur la structure hiérarchique de Wikipédia et ses données structurées (système de catégories de Wikipédia, infobox, etc) [23], mais il existe aussi des travaux visant à extraire les types à partir de textes en ayant recours à des règles sur les catégories grammaticales (Part-of-speech, POS) [24], sur les patrons syntaxiques [25], sur des techniques d'apprentissage machine [26], ou encore des systèmes hybrides combinant les techniques reposant sur des patrons lexico-syntactiques et des techniques de classification automatique [27] [28].
- **Relations non hiérarchiques:** Cette couche permet la description des instances et les relations les reliant. Certains travaux, comme [29], utilisent les données structurées de Wikipédia pour extraire ce type de relations comme par exemple la propriété birthdate dans l'infobox d'une personne. D'autres travaux ont recours à des méthodes de traitement automatique de la langue pour extraire ces relations : on compte TextRunner [13] ou encore WOE [14] comme exemples de systèmes utilisant l'apprentissage statistique. PORE [30] fait aussi appel à des techniques d'apprentissage machine pour l'extraction d'informations de Wikipédia.
- **Axiomes et règles logiques:** L'extraction et la définition des axiomes et règles logiques est le champ de recherche le moins exploité parmi les six couches. L'importance des axiomes vient du fait qu'ils permettent d'effectuer des raisonnements logiques et ainsi inférer de nouvelles connaissances dans une ontologie. Comme expliqué à la section 2.3.2, parmi les travaux se concentrant sur la définition d'axiomes et de règle logiques, on compte [31], [32], [33].

Il existe dans la littérature des recherches qui visent à enrichir chacune des couches du *ontology layer cake*, cependant, il n'existe pas de travaux aboutissant à la création d'une ontologie riche qui couvre l'ensemble de ces couches. De plus, malgré les efforts pour enrichir DBpedia [34] [35], il n'existe pas à notre connaissance de travaux permettant de la doter d'axiomes. Le but de ce mémoire est de générer automatiquement une ontologie plus expressive, qui offrira une plus grande capacité d'inférence.

2.3 Travaux reliés à notre recherche

Notre travail de recherche repose sur l'extraction d'information à partir du texte contenu dans Wikipédia, et vise à enrichir DBpedia et à étendre son ontologie. Nous contribuons essentiellement à deux couches du *ontology layer cake*; la couche de taxonomie et la couche d'axiomes et de règles logiques. Notons toutefois que le fait de définir des axiomes comprenant des classes, des opérateurs logiques et des propriétés fait que nous contribuons également indirectement aux trois autres couches. Nous nous concentrons d'abord sur l'extraction de trois types d'information : la nature des entités (la distinction entre classes ou instances) (2.3.1), les axiomes et les règles logiques définissant les classes (2.3.2), les types des instances (2.3.3). Finalement nous avons recours à la désambiguïsation de classes (2.3.4). Dans cette section, nous présentons un bref état de l'art pour chacune des sous-tâches de notre travail, qui seront ensuite détaillées dans les Chapitres 4, 5, et 6.

2.3.1 Distinction entre classes et instances

Une classe étant un concept du monde et une instance étant une entité unique du monde, il est facile pour un être humain de faire la distinction entre des classes et instances. Cependant, il s'agit d'une tâche qui n'est pas évidente à automatiser, et il existe peu de recherches, comme [36], aboutissant à des résultats tangibles pour cette tâche. Nous expliquons plus en profondeur, dans le chapitre 6, l'importance de bien distinguer et séparer classes et instances, et nous y présentons plus en détail les recherches ayant ce sujet pour objectif.

2.3.2 Extraction d'axiomes définissant des classes

Pour qu'une ontologie soit vraiment utile, il est important de définir des axiomes logiques définissant les classes qu'elle contient. Définir les classes d'une base de connaissances permet d'augmenter l'expressivité de la base de connaissance afin de pouvoir inférer de l'information. Pour extraire des axiomes, il existe des recherches utilisant des approches basées sur des patrons [32] et des approches statistiques [33]. L'état de l'art visant cette tâche est présenté plus en détail dans les articles 1 et 3 (chapitre 4 et 6).

Nous avons adopté une approche basée sur les patrons syntaxiques et grammaticaux pour l'extraction d'axiomes à partir de textes. De plus, nous avons fourni un effort particulier pour

désambiguïser les classes créées dans les axiomes en les associant à des URIs de DBpedia; nous obtenons donc une ontologie intégrable à DBpedia.

2.3.3 Extraction de types d'instances

Pour les instances, il est essentiel qu'elles soient reliées aux classes auxquelles elles appartiennent dans une base de connaissances. Pour cette étape, nous avons recours à une approche [37] recherchant les patrons syntaxiques dans une phrase. Il existe des recherches qui se basent sur les informations extraites de textes, à partir des catégories grammaticales [24], de la recherche de patrons syntaxiques [25] ou de méthodes d'apprentissage machine [26] pour typer les instances. Cependant, puisque le traitement du texte peut s'avérer plus complexe, une plus grande partie des relations de typage dans DBpedia sont extraites des données structurées de Wikipédia [23]. Par ailleurs, certaines méthodes se concentrent sur les connaissances déjà existantes pour inférer de nouvelles connaissances et extraire des types manquant dans les bases de connaissances. Cependant, les données bruitées sont récurrentes dans les larges bases de connaissances, et sont source d'erreurs et d'extraction de fausses connaissances. L'approche SDType [38] exploite les relations entre instances pour inférer leurs types, et tolère le bruit en ayant recours à des méthodes de votes pondérés. Certains travaux combinent différentes méthodes afin d'obtenir de meilleurs résultats [27] [28]. Les articles 2 et 3 (chapitre 5 et 6) présentent plus en détail certaines recherches ayant pour objet le typage d'instances.

2.3.4 Désambiguïsation

Afin d'intégrer notre base de connaissance, AXIOpedia, à DBpedia, il a fallu avoir recours à la désambiguïsation pour effectuer des liens à partir des URIs de DBpedia. Notre approche pour ce sujet est une approche préliminaire qui se base uniquement sur les chaines de caractères. Dans [37], présenté dans le chapitre 5, nous avons recours à un pipeline de méthodes pour la désambiguïsation (Figure 2.3) ; tout d'abord nous avons recours à une recherche d'URI à partir de la chaine de caractères, par exemple, pour le type *oke:Villain* que nous voulons désambiguïser, on recherche l'URI *dbo:Villain*. Si cette URI n'existe pas, nous avons recours à différentes méthodes, en utilisant les relations de DBpedia; la méthode suivante se base sur les types informels des instances, par exemple, si nous obtenons le type *oke:Village* et que *dbo:Village* n'existe pas, mais que *dbr:Village* existe, nous utilisons ce type pour trouver d'autres instances ayant ce même type en plus d'un type

formel, par exemple, *dbr:Bogoria,_Poland* contient le type *dbr:Village*, mais aussi le type *dbo:Place* auquel nous alignons *oke:Village*. La prochaine méthode du pipeline se base sur les autres types de l'instance ; nous cherchons une relation *rdf:type* pour l'entité en entrée, par exemple si nous avons extrait *dbr:Adalgis rdf:type oke:King*, et que le triplet *dbr:Adalgis rdf:type dul:NaturalPerson* existe sur DBpedia, cela impliquerait que *oke:King* pourrait s'aligner à *dul:NaturalPerson*. Si ces méthodes n'aboutissent pas, nous nous référons aux types informels de l'instance et nous recherchons leurs types directs, par exemple pour aligner *oke:Club*, nous recherchons *dbr:Club* et les relations *rdf:type* de cette ressource. La prochaine méthode utilise les catégories de la ressource, par exemple pour *dbr:Village*, nous avons les catégories *dbc:Administrative_divisions*, *dbc:Villages*, etc., nous recherchons par la suite les types des ressources dans chacune de ces catégories pour effectuer l'alignement, par exemple *dbc:Villages* contient plusieurs instances (tel que *dbr:Mallekan)* ayant pour type *dbo:Place; dbo:Place* est alors utilisé pour la désambiguïsation. Finalement, nous nous basons sur les *domain* et *range* des relations de l'instance pour inférer son type; par exemple, à partir des relations *dbr:Marko Vovchok dbo:birthplace dbr:Village* et *dbo:birthPlace rdfs:range dbo:Place* nous pourront inférer à partir du *range* de la relation *dbo:birthplace* que *dbr:Village* est de type *dbo:Place*. Toutes ces méthodes sont détaillées dans le chapitre 5.

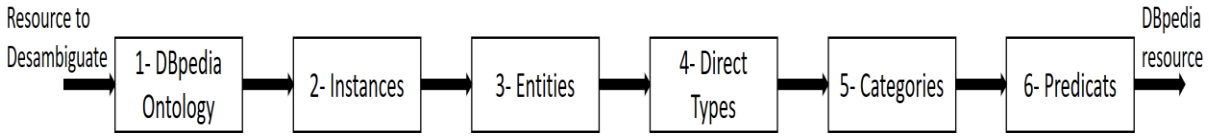

Figure 2.3 Pipeline des méthodes de désambiguïsation

D'autres travaux, comme DBpedia Spotlight [39] ont recours à la désambiguïsation d'entité, et utilisent un modèle vectoriel pour la représentation des ressources de DBpedia, et le calcul de scores permettant la classification des ressources en fonction de la similarité de contexte. Des travaux plus poussés dans ce domaine font appel à des systèmes complexes [39] qui utilisent les connaissances de Wikipédia, pour construire un *réseau sémantique à grande échelle* pour la désambiguïsation d'entités nommées. Par ailleurs, [40] repose sur la création de graphes pondérés

en se basant sur de larges bases de connaissances comme YAGO, et en se concentrant sur trois mesures: la probabilité qu'une entité soit mentionnée, les similarités de contextes, et la cohérence entre les entités candidates. D'autres méthodes, comme [41], cherchent à extraire le contexte de l'entité à partir de la hiérarchie des catégories comme référence pour la désambiguïsation. Toutes ces techniques d'annotation sémantique de textes pourraient ultimement être adaptées à la désambiguïsation d'axiomes. Cela représente toutefois un travail futur non couvert par ce mémoire.

CHAPITRE 3 MÉTHODOLOGIE

Ce chapitre aborde la méthodologie de recherche suivie dans ce mémoire. Dans la section 3.1, nous discutons des patrons syntaxiques et expliquons comment nous les avons établis. Dans la section 3.2, nous décrivons le rôle de SPARQL et de RDF dans notre travail. Nous présentons par la suite, dans la section 3.3, l'architecture générale de notre pipeline. La section 3.4 s'attarde sur l'ontologie AXIOpedia construite. La section 3.5 présente le service Web que nous offrons pour accéder à notre système. Finalement, dans la section 3.6, nous présentons et discutons de la phase d'évaluation.

3.1 Patrons syntaxiques

L'identification des patrons syntaxiques est une étape de prétraitement. Nous nous sommes basés sur des définitions extraites de Wikipédia pour identifier les patrons syntaxiques les plus fréquents. Cette étape ayant été manuelle, on suppose qu'il serait encore possible de détecter des patrons syntaxiques plus complexes, ou moins fréquents, en automatisant cette tâche. Nous avons par la suite associé chacun de ces patrons à une procédure CONSTRUCT de SPARQL pour la construction d'un triplet ou un ensemble de triplets OWL lors de la présence de ce patron. La liste complète est constituée de 32 patrons pour la définition d'axiome et 4 patrons pour la détection d'instances *rdf:type*, en plus des 4 patrons de Hearst, présentés respectivement dans les tables 6.13, 6.14 et 6.15 dans l'annexe de l'article 3.

3.2 SPARQL et graphe RDF

Une innovation de notre travail de recherche est l'utilisation de SPARQL pour la recherche de patrons dans les graphes RDF.

Afin de traiter une phrase, nous effectuons tout d'abord une analyse syntaxique de celle-ci en ayant recours au Stanford parser [42]. À partir de cette analyse, nous construisons un graphe RDF représentant cette phrase, ainsi que les relations lexicales et grammaticales entre les mots de la phrase. Ce graphe servira par la suite de support pour faciliter la recherche d'occurrence de patrons syntaxiques dans la définition.

Pour la recherche des patrons, nous avons implémenté chacun des patrons sous forme d'une unique requête SPARQL capable de détecter ce patron dans un graphe RDF, et de construire les axiomes qui lui sont associés.

La motivation derrière le choix de SPARQL est la simplicité de ce langage, en plus du fait qu'il s'agit de l'une des technologies du Web sémantique. Afin de retrouver les occurrences d'un patron, notre postulat de départ était qu'il suffisait d'implémenter une unique requête SPARQL et qu'il serait donc relativement facile d'ajouter un nouveau patron. Toutefois, nous avons découvert certaines limites de SPARQL, telles que la récursivité et la manipulation de liste. En effet, il est difficile de construire des listes avec un nombre non déterminé de membres, ou de manipuler les membres d'une liste dont on ne connait pas spécifiquement les URIs. Même si ces aspects ont rendu plus difficile l'implémentation, nous avons réussi à faciliter le problème en divisant les patrons SPARQL à exécuter en deux groupes. La première étape s'occupe de l'agrégation et la création de listes et est constituée de requêtes SPARQL plus complexes, qui modifient le graphe pour faciliter le reste du travail. La deuxième phase est constituée de l'ensemble des patrons syntaxiques implémentés par de simples requêtes SPARQL. Afin d'ajouter de nouveaux patrons, il suffit de créer des requêtes SPARQL à ajouter dans la seconde étape, sans modifier la première.

3.3 Architecture générale

Nous avons ainsi conçu un pipeline qui prend en entrée une définition en anglais, et fournit en sortie un ensemble de triplets représentant l'information extraite de la définition. On divise notre pipeline en 5 modules principaux illustrés à la figure 2, qui décrit l'architecture générale de notre système.

Le premier module consiste à traiter la définition textuelle en entrée, et à effectuer l'analyse syntaxique pour créer le graphe RDF la représentant. Ce graphe sert de base à notre travail, et facilite par la suite la recherche d'occurrences des patrons syntaxiques implémentés à partir de requêtes SPARQL.

Le second module sert à distinguer les pages Wikipédia et à les classer en deux groupes : le premier contient les pages représentant des classes, et le second contient les pages qui représentent des instances. La raison de la séparation du traitement des classes et des instances est que, dans une ontologie, les classes et les instances ne sont pas définies de la même façon; tout d'abord, les triplets à construire ne sont pas les mêmes selon la nature des pages, et de plus on ne voudrait pas extraire

la même information si on fait face à une classe ou une instance, puisque le type d'informations pertinentes à extraire pour une classe n'est pas le même que pour les instances.

En effet, lorsque nous traitons une classe dans une ontologie, il est utile d'avoir un axiome définissant la classe. En effet, cette information aide à documenter les classes, à détecter des inconsistances dans l'ontologie, et à inférer de l'information sur les instances de la classe. Par exemple, pour une définition de classe comme *A poet is a person who writes poetry*, notre objectif est d'extraire l'axiome $Poet \equiv Person \cap \exists writes.Poetry$; cette définition révèle qu'un poète est une personne pour laquelle il existe une relation «écrire un poème». Le processus détaillé pour extraire cet axiome est présenté dans les articles 1 et 3 (chapitre 4 et 6).

Dans le cas d'une instance, nous voulons extraire le type de celle-ci, et ainsi la lier à une ou plusieurs classes. Par exemple, pour la définition de l'instance *Charles Pierre Baudelaire* qui est *Charles Pierre Baudelaire was a French poet who also produced notable work as an essayist, art critic, and pioneering translator of Edgar Allan Poe*, notre objectif est d'extraire les triplets *CharlesBaudelaire rdf:type FrenchPoet*, *CharlesBaudelaire rdf:type Essayist*, *CharlesBaudelaire rdf:type ArtCritic*, *CharlesBaudelaire rdf:type Translator*. Les étapes de la méthodologie pour extraire ces triplets sont détaillées dans les articles 2 et 3 (chapitre 5 et 6).

Ainsi, selon la nature de la page, nous appelons le module 3 ou 4, qui traite le graphe RDF pour en extraire un axiome si la page représente une classe (module 3 à gauche de la Figure 2), ou pour en extraire le type (module 4 à droite de la Figure 2), si elle représente une instance.

Aussi, selon la nature de la page (classe ou instance), nous exécutons un pipeline de requêtes SPARQL différent sur le graphe RDF construit, afin d'extraire les triplets pertinents de la phrase. La liste des patrons à exécuter dans le cas d'une classe est présentée dans la table 6.13 de l'annexe de l'article 3, et la liste des patrons à exécuter dans le cas d'une instance est présentée dans les tables 6.14 et 6.15 de cette annexe.

Finalement, nous avons recours au dernier module qui s'occupe de la désambiguïsation, afin de faciliter l'intégration de la sortie du système à DBpedia. L'ensemble des triplets construits sur l'ensemble des pages Wikipédia est stocké dans notre ontologie, AXIOpedia, qui fait référence à DBpedia grâce à l'étape de désambiguïsation.

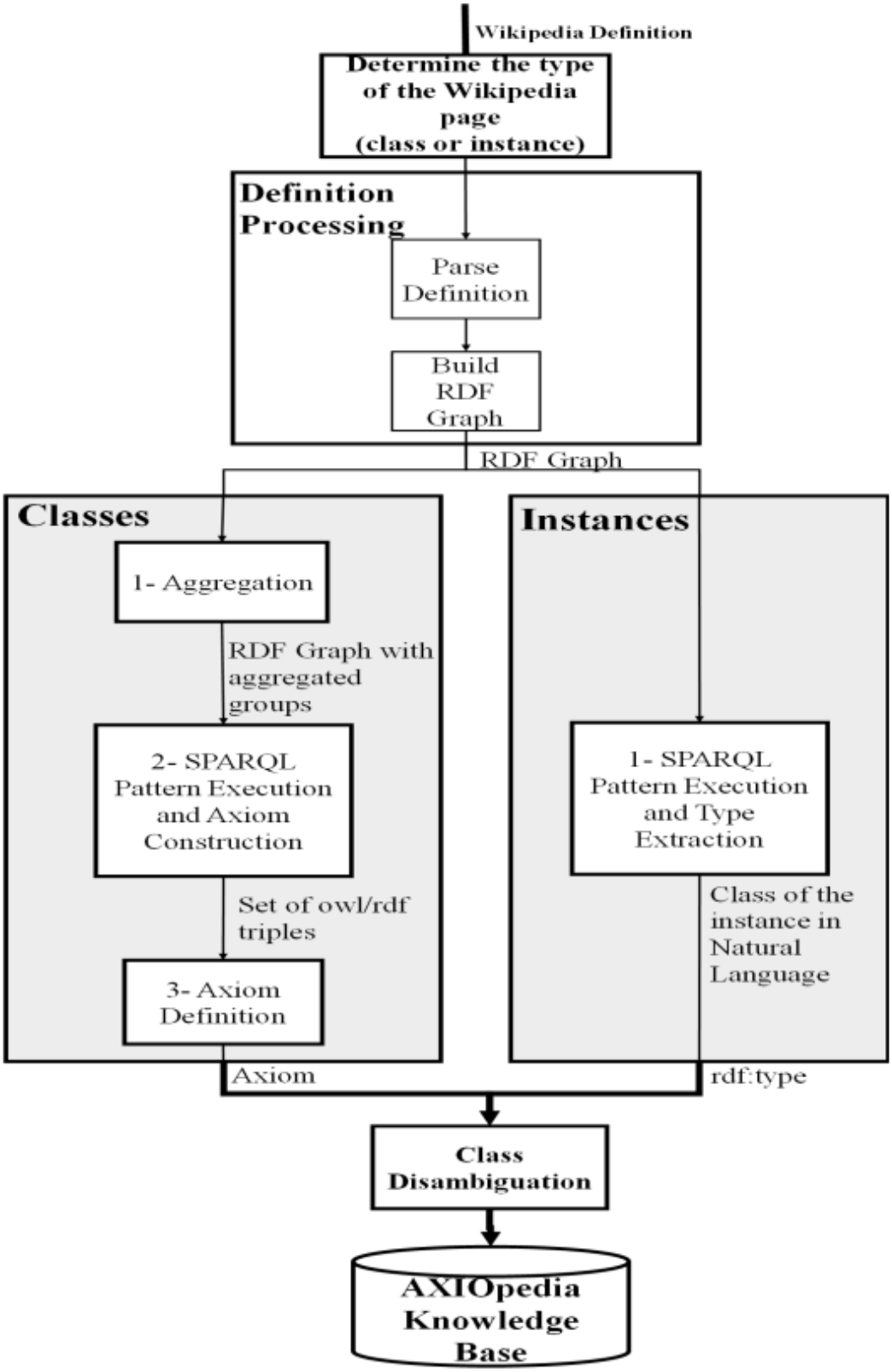

Figure 3.1 Architecture générale

3.4 Base de connaissances AXIOpedia

Comme expliqué dans le premier chapitre, notre principal objectif est d'enrichir DBpedia avec des axiomes définissant les classes ainsi que des liens taxonomiques permettant de typer des instances. Pour ce faire, nous avons parcouru l'ensemble des pages de Wikipédia et traité la définition de chacune de ces pages afin d'en extraire les informations pertinentes. Nous avons alors exécuté le pipeline sur chacune des pages pour construire une base de connaissances commune, AXIOpedia.

Nous obtenons donc une base de connaissance contenant des axiomes définissant les classes, des instances typées et reliées à leurs classes, ainsi que des liens taxonomiques et de nouvelles classes générées. Grâce à l'étape de désambiguïsation, AXIOpedia fait référence aux URIs de DBpedia et s'intègre donc à DBpedia.

L'intégration d'AXIOpedia à DBpedia, contribue à l'enrichissement de DBpedia notamment grâce à l'ontologie d'AXIOpedia qui est beaucoup plus expressive.

3.5 Service Web

Nous nous sommes posés comme objectif d'offrir un service Web[4] permettant de donner accès à notre outil aux chercheurs voulant extraire des axiomes et liens taxonomiques à partir d'une définition textuelle. À partir de ce service, nous donnons aux utilisateurs le choix entre deux options; utiliser l'outil pour sur une définition textuelle au choix, ou, consulter et naviguer dans l'ontologie AXIOpedia construite à partir des définitions extraites de Wikipédia en Janvier 2017.

3.6 Présentation des articles

Comme mentionné plus haut, nous présentons trois articles dans les chapitres 4, 5 et 6; Le premier article *Automatic Extraction of Axioms from Wikipedia Using SPARQL*, publié dans le cadre de la conférence *Extended Semantic Web Conference (ESWC)* en 2016. Le deuxième article *Entity Typing and Linking Using SPARQL Patterns and DBpedia*, publié dans le cadre de la compétition Open Knowledge Extraction (OKE) de la conférence *Extended Semantic Web Conference (ESWC)*

---

[4] www.westlab.polymtl.ca/AXIOpediaWebApp/Home

en 2016. Et le troisième article *AXIOpedia: Enriching DBpedia with OWL Axioms from Wikipedia* soumis au *Semantic Web Journal* en avril 2017.

Le premier article présente le traitement des classes, alors que le second présente le traitement des instances. Nous avons mis le travail en commun afin de pouvoir créer une base de connaissances complète contenant à la fois les informations concernant les classes et celles concernant les instances. Pour y arriver, nous avons ajouté une étape de prétraitement capable de détecter le type de la page Wikipédia traitée, et distinguer classes et instances afin de traiter la définition convenablement. Ainsi l'article 3 présente le pipeline complet incluant le travail effectué dans chacun des articles.

3.7 Évaluation

Afin de juger la qualité de l'ontologie AXIOpedia, nous avons évalué séparément les composantes de notre système. Plus précisément, nous avons évalué séparément le filtrage des pages en classes et instances, les axiomes construits pour les classes, les types extraits pour les instances, et la désambiguïsation.

Pour chacune de ces parties, nous avons eu recours à une évaluation manuelle, en plus d'autres évaluations dans certains cas. Pour les évaluations manuelles, nous avons eu recours à six étudiants du laboratoire WeST séparés en trois groupes de deux évaluateurs. Chaque groupe a eu à annoter un ensemble de données différent.

Tout d'abord, pour le filtrage des pages (en classes ou instances) et la désambiguïsation des URIs, nous avons uniquement eu recours à des évaluations manuelles. Nous avons calculé les valeurs d'exactitude pour l'évaluation du filtrage des pages et les valeurs de précision et rappel pour l'évaluation de la désambiguïsation. Pour le filtrage des pages, on aboutit à une exactitude de 88%, alors que pour la désambiguïsation des URIs, on aboutit à une précision de 72% et un rappel de 41%.

En ce qui concerne les axiomes, il n'existe pas de mesures ou de règles précises pour une telle évaluation. Nous avons donc mis en place un score indicateur de la qualité d'un axiome extrait d'une phrase.

Pour chaque axiome, l'évaluateur construit l'axiome correct et le compare à l'axiome généré afin de spécifier le nombre de classes, prédicats, et opérateurs qui sont corrects, incorrects, et

manquants. Nous calculons par la suite la précision et le rappel pour chacun des composants (classes, prédicats et opérateurs) et en effectuons la moyenne. Ainsi, nous obtenons des scores moyens de 76% pour la précision et 74% pour le rappel.

Pour les types d'instances, nous avons tout d'abord eu recours à une évaluation manuelle, dans laquelle nous avons calculé les valeurs de précision et rappel ; nous obtenons 86% pour la précision et 75% pour le rappel. Nous avons aussi eu recours à l'évaluation effectuée par la compétition OKE pour laquelle notre approche a été l'approche gagnante pour la tâche s'occupant de l'extraction de types pour l'année 2017. Nous avons obtenu des micros et macros précisions et rappels entre 73% et 82%. En plus, nous avons effectué une dernière évaluation basée sur DBpedia, dans laquelle nous comparons les types que nous avons extraits aux types déjà présents dans DBpedia ; dans cette évaluation, nous étions capables de typer 43% des pages DBpedia qui n'avaient aucun type.

L'ontologie étant encore en construction, il n'est pas possible de fournir des statistiques sur ses composantes ; ces informations seront disponibles sur notre site web[5] lorsque l'ontologie sera complètement générée.

[5] www.westlab.polymtl.ca/AXIOpediaWebApp/Home

CHAPITRE 4 ARTICLE 1: AUTOMATIC EXTRACTION OF AXIOMS FROM WIKIPEDIA USING SPARQL

Lara Haidar-Ahmad, Amal Zouaq, Michel Gagnon: Automatic Extraction of Axioms from Wikipedia Using SPARQL. ESWC (Satellite Events) 2016: 60-64.

In: The Semantic Web - ESWC: International Semantic Web Conference.

Abstract. Building rich axiomatic ontologies automatically is a step towards the realization of the Semantic Web. In this paper, we describe an automatic approach to extract complex classes' axioms from Wikipedia definitions based on recurring syntactic structures. The objective is to enrich DBpedia concept descriptions with formal definitions. We leverage RDF to build a sentence representation and SPARQL to model patterns and their transformations, thus easing the querying of syntactic structures and the reusability of the extracted patterns. Our preliminary evaluation shows that we obtain satisfying results, which will be further improved.

4.1 Introduction

Building rich ontologies with reasoning capabilities is a difficult task, which can be time consuming. It requires both the knowledge of domain experts and the experience of ontology engineers. This is one of the main reasons why current Semantic Web and linked data rely mostly on lightweight ontologies. The automatization of axiom extraction is a step towards creating richer domain concept descriptions [43] and building a Semantic Web that goes beyond explicit knowledge for query answering. Ontology learning, i.e. the automatic extraction of ontologies from text, can help automatize the extraction of primitive, named and complex classes. Few state of the art approaches were developed to achieve this goal, mostly pattern-based approaches [44] [3]. To our knowledge, LExO [45] is the most advanced system for complex class extraction. This paper describes our approach to extract defined and primitive class axioms from Wikipedia concept definitions using SPARQL. The main contribution of this work is i) the utilization of SPARQL graph matching capabilities to model patterns for axiom extraction ii) the description of SPARQL

patterns for complex class extractions from definitions and iii) The enrichment of DBpedia concept descriptions using OWL axioms and defined classes. We also briefly compare our preliminary results with those of LEXO.

4.2 Methodology

We rely on a pattern-based approach to detect syntactic constructs that denote complex class axioms. These axioms are extracted from Wikipedia definitions.

Definition Representation and General Pipeline: We process definition sentences and first construct an RDF graph that represents the dependency structure of the definition and the words' part of speech and positions in the sentence. This step makes the subsequent step of pattern matching using SPARQL requests easier. For every word, we specify its label, its part-of-speech, its position in the sentence and its grammatical relations with the other words based on the output of the Stanford parser [42]. Figure 4.1 presents an example of the RDF graph of a definition. For this example, we use the definition of the Wikipedia concept *Vehicle* from our dataset, which is "Vehicles are non-living means of transportation".

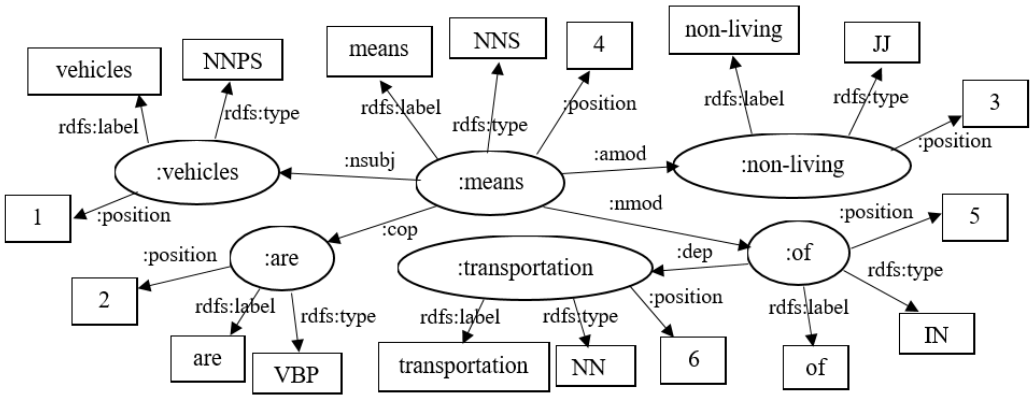

Figure 4.1 The RDF Representation of the definition of vehicles.

Based on this RDF representation, we execute a pipeline of SPARQL requests on the obtained RDF graphs (see Figure 4.2).

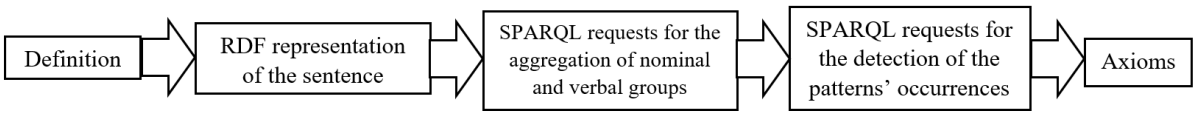

Figure 4.2 Pipeline

First, we execute SPARQL aggregation requests to extract complex expressions such as nominal and verbal groups and define subclass axioms. For instance, for the sentence *vehicles are non-living means of transportation*, we obtain the following expressions: *vehicles*, *non-living means of transportations* and *means of transportations*. We also extract the axiom *subClassOf* (*Non-living means of transportation, Means of transportation*). Finally, we execute a set of SPARQL axiom queries to identify occurrences of patterns that can be mapped to OWL complex class definitions.

SPARQL Pattern Representation: Based on a randomly chosen set of 110 definitions from Wikipedia and their sentence representation, we identified several recurring syntactic structures manually and built their corresponding SPARQL patterns. Next, we mapped patterns to complex class axioms using SPARQL CONSTRUCT. Table 4.1 presents the most common patterns that we identified in our dataset, in addition to their corresponding axioms. Each pattern is modeled using a single SPARQL request. This mechanism provides simple ways to enrich our approach with patterns that we do not support yet.

Table 4.1 Most Frequent Patterns and their Respective Axioms

Frequent patterns for the definitions of concepts	Corresponding Axioms
(1) SUBJ copula COMP *Vehicles are non-living means of transportation.*	SUBJ $\subseteq$ COMP *Vehicles* $\subseteq$ *NonLivingMeansOfTransportations*
(2) SUBJ copula COMP that VERB OBJ *A number is an abstract entity that represents a count or measurement*	SUBJ $\equiv$ (COMP $\cap$ $\exists$VERB.OBJ) *Number* $\equiv$ *(AbstractEntity* $\cap$ $\exists$*represents.(Count* $\cup$ *Measurement))*
(3) SUBJ copula COMP VERB preposition NOUN *A lake is a body of water surrounded by land.*	SUBJ $\equiv$ (COMP $\cap$ $\exists$VERB_prep.NOUN) *Lake* $\equiv$ *(BodyOfWater* $\cap$ $\exists$*surroundedBy.Land)*

4.3 Preliminary Evaluation and Discussion

We compared the generated axioms with a manually-built gold standard containing 20 definitions chosen randomly from our initial dataset[6]. We assessed the correctness of the axioms using standard precision and recall by focusing on named classes, predicates and complete axioms (see Table 4.2). Complete axioms metrics are calculated by counting the number of classes, predicates and logical operators matched with the ones in the gold standard. We obtain a macro precision and recall of 0.86/0.59 respectively. We also propose an axiom evaluation based on the Levenstein similarity metric which considers each axiom as a string. The higher the Levenstein similarity between the generated axiom and the reference, the most similar the axioms are. We tested multiple similarity levels as shown in Table 4.3. We notice that we usually generate the right axioms for i) small sentences ii) sentences with a simple grammatical structure and iii) longer sentences which have no grammatical ambiguities. We also notice that false positives are rarely generated, and the errors in our results are usually caused by incomplete axioms. This is explained by the limited number of implemented patterns (10 patterns).

Table 4.2 Evaluation Results

	Classes		Predicates		Complete Axioms	
	Precision	Recall	Precision	Recall	Precision	Recall
Macro	0.87	0.66	0.94	0.54	0.86	0.59
Micro	0.86	0.61	0.76	0.36	0.78	0.48

Table 4.3 Axioms' Precision based on Levenstein Similarity with the Gold Standard

Similarity Level	0.70	0.80	0.90	1.00
Levenstein Precision	0.55	0.50	0.35	0.30

[6] The dataset and gold standard are available at: http://westlab.herokuapp.com/axiomfactory/dataESWC16

While LExO [46] adopted a similar approach to ours, they did not rely on standard Semantic Web languages such as SPARQL for their patterns and did not take into account the aggregation of nominal and verbal groups, or the extraction of taxonomical relations. For example, given the definition *A minister or a secretary is a politician who holds significant public office in a national or regional government*, LExO generates $(Minister \cup Secretary) \equiv (Politician \cap \exists holds.((Office \cap Significant \cap Public) \cap \exists in.(Government \cap (National \cup Regional))))$. In contrast, our system generates the axiom $Minister \equiv Secretary \equiv (Politician \cap \exists\ holds.(SignificantPublicOffice \cap \exists in.(NationalGovernment \cup RegionalGovernment)))$, and in addition, it generates a taxonomy where, *SignificantPublicOffice* is a subclass of *PublicOffice*, and *NationalGovernment* and *RegionalGovernment* are subclasses of *Government*.

4.4 Conclusion and Future Work

The paper describes an approach to extract OWL axioms with the aim to *logically define* DBpedia concepts from Wikipedia definitions using SPARQL requests. We are currently working on the implementation of our pipeline as a Web service, which has not been proposed yet in the state of the art. More importantly, one original contribution of this paper is the reliance on Semantic Web languages (RDF, SPARQL) to model sentences, patterns and axioms, thus easing the reusability and enrichment of the defined patterns.

4.5 Acknowledgement

This research has been funded by the NSERC Discovery Grant Program.

4.6 References

De Marneffe, M-C, and Manning, C. D. (2008). The Stanford typed dependencies representation. Proc. of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation. ACL.

Bühmann, L., Fleischhacker, D., Lehmann, J., Melo, A. & Völker, J. (2014). Inductive Lexical Learning of Class Expressions. In Knowledge Engineering and Knowledge Management (pp. 42-53). Springer International Publishing.

Völker, J., Haase, P. and Hitzler, P. (2008) Learning expressive ontologies, in: Proc. of the Conference on ontology Learning and Population, IOS Press, 2008, pp. 45–69

Font, L., Zouaq, A., Gagnon, M. (2015). Assessing the Quality of Domain Concepts Descriptions in DBpedia. SITIS 2015: 254-261

CHAPITRE 5 ARTICLE 2: ENTITY TYPING AND LINKING USING SPARQL PATTERNS AND DBPEDIA

Lara Haidar-Ahmad, Ludovic Font, Amal Zouaq, Michel Gagnon: Entity Typing and Linking Using SPARQL Patterns and DBpedia. SemWebEval@ESWC 2016: 61-75.

In: Semantic Web Challenges - Third SemWebEval Challenge at ESWC 2016.

Abstract. The automatic extraction of entities and their types from text, coupled with entity linking to LOD datasets, are fundamental challenges for the evolution of the Semantic Web. In this paper, we describe an approach to automatically process natural language definitions to a) extract entity types and b) align those types to the DOLCE+DUL ontology. We propose SPARQL patterns based on recurring dependency representations between entities and their candidate types. For the alignment subtask, we essentially rely on a pipeline of strategies that exploit the DBpedia knowledge base and we discuss some limitations of DBpedia in this context.

5.1 Introduction

The growth of the Semantic Web depends on the ability to handle automatically the extraction of structured information from texts and the alignment of this information to linked datasets. The first OKE Challenge competition [46] targeted these two issues and is a welcome initiative to advance the state of the art of open information extraction for the Semantic Web. In this paper, we present our service for entity typing and linking using SPARQL patterns and DBpedia[7]. This service is the winner of the OKE challenge 2016 Task 2.

---

[7] http://westlab.polymtl.ca/OkeTask2/rest/annotate/post

Besides a participation to the OKE challenge, one aim of this research is to provide a task-based evaluation of the DBpedia knowledge base. Hence our linking strategies exploit both the DBpedia ontology and the DBpedia knowledge base to extract *rdfs:subClassOf* relationships between natural language types and DBpedia types.

This paper is structured as follows: Section 2 presents some related work. Sections 3 and 4 describe the two subtasks of our service: type recognition and extraction from text, and type alignment using the ontology Dolce+DUL. In Section 5, we present the evaluation of our system. We discuss our results in Section 6.

5.2 Related Work

Several tasks are related to the challenge of entity typing and alignment, among which we can cite named entity recognition [47], relation extraction [48] [49] [50], ontology learning [51] and entity linking [52] [53] [54]. Due to space constraints, this state of the art will be limited to the participants of the previous OKE challenge [46].

5.2.1 Type extraction

The automatic extraction of *taxonomical* and *instance-of* relations from text has been a long-term challenge. Overall, state-of-the-art approaches that target the extraction of relations from text are mainly pattern-based approaches. In the first edition of the 2015 OKE challenge, there were three participating systems for the task of type extraction from natural language definitions: CETUS [24], OAK@Sheffield [26] and FRED [27]. CETUS relies on grammar rules based on parts of speech (POS) to extract an entity type from text. OAK uses machine learning to learn to recognize the sentences' portions that express the entity type, and then uses a POS pattern grammar for type annotation. FRED uses the system Boxer [55] and Discourse Representation Theory, and thus relies on a complex architecture for ontology extraction that is not limited to type extraction. Compared to previous pattern-based approaches in the OKE competition [24] [56], our system differs by the nature of the patterns, which exploit a dependency grammar representation. One particular novelty is the use of SPARQL to model and search for patterns occurrences. Overall, we believe that our approach represents a middle ground between patterns based on a superficial representation of sentences (usually parts of speech) and approaches such as FRED [27] which depend on complex first-order logic and frame semantics.

5.2.2 Type alignment

In the context of the Semantic Web, the challenge of entity typing is coupled with the difficulty of finding an alignment with linked datasets. Among the three systems of the OKE challenge 2015 mentioned previously, the authors of CETUS [24] developed an alignment between Yago and Dolce + DnS Ultralite; FRED [27] uses an already existing API that exploits Dolce, WordNet and VerbNet; OAK [26] relies on the existence of *dul* types in DBpedia, using a method similar to our method 2 (see Section 4.1). In our approach, we chose to use the existing mappings DBpedia - Dolce + DnS Ultralite [57] and Yago wordnet - Dolce + DnS Ultralite [58].

Our main contribution in this subtask is the exploitation of several strategies that consider either the DBpedia ontology (T-box) or the DBpedia knowledge base (A-box) to find a DBpedia type. We exploit both the knowledge about the entity and the type given as input. When there is not any direct type information linked to the DBpedia ontology or Yago, we revert to type inference methods. Among the strategies described in section 4.1, method 6 is based on our previous work [59] to infer types using predicates' domain and range, while method 2 is similar to the one used by OAK [26]. However, we also introduce a novel approach based on DBpedia categories and propose a pipeline of strategies that aggregates several methods.

5.3 Entity Type Extraction

Entity type extraction consists in finding the natural language type of an entity, given its textual definition. Our approach relies on pattern extraction using a dependency-based syntactic analysis. The extraction of an entity type is processed in two steps: sentence representation in RDF and pattern occurrence identification using SPARQL queries.

5.3.1 Sentence Graph Representation

First, we extract grammatical dependencies from the definitions using the Stanford parser [42].and build an RDF graph representing each sentence. Before the parsing step, we identify the input DBpedia entity in the sentence and aggregate multi-words entities with an underscore between the words. For instance, in the sentence *All's Well That Ends Well is a play by William Shakespeare*, we identify *All's Well That Ends Well (*the input DBpedia resource) as one single entity and simply modify the sentence to obtain *All's_Well_That_Ends_Well is a play by William Shakespeare*.

We then construct an RDF graph representing the dependency structure of the definition. Thus we specify the label and part of speech of each word in addition to its grammatical relations with the other words. This RDF graph allows us to look for pattern occurrences using SPARQL requests in the following step. Figure 5.1 presents the RDF graph of the definition *Skara Cathedral is a church in the Swedish city of Skara*.

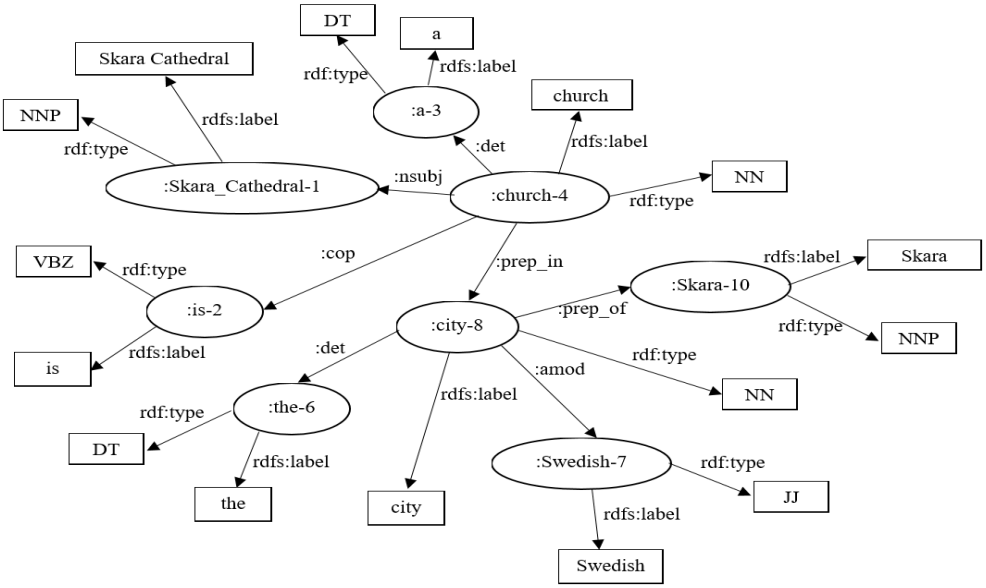

Figure 5.1 The RDF Representation of the definition of Skara Cathedral

5.3.2 Pattern Identification

As for the detection of patterns, based on the train dataset[8] distributed in the OKE challenge, we manually identified several recurring syntactic and grammatical structures between the entities and their respective types. Table 5.1 presents the most common patterns that we identified in the dataset.

---

8 https://github.com/anuzzolese/oke-challenge-2016/blob/master/GoldStandard_sampleData/task2/dataset_task_2.ttl

Table 5.1 Most Frequent Patterns Describing an Entity/Type Relationship

Frequent patterns
(1) :nsubj :cop Entity X Z Type :prep_of Where X = { “name”, “nickname”, “alias”, “one”, “species”, “form” } *Sant'Elmo is the name of both a hill and a fortress in Naples, located near the Certosa di San Martino.* ***Entity** = Sant'Elmo ; **Type** = Hill*
(2) :nsubj :cop Entity Type Z *El Oso, released in 1998, is an album by the New York City band Soul Coughing.* ***Entity** = El Oso ; **Type** = Album*
(3) :nsubjpass :auxpass Entity V Z Type :prep_as *Bromius in ancient Greece was used as an epithet of Dionysus/Bacchus.* ***Entity** = Bromius ; **Type** = Epithet*
(4) x Entity Type Where x = { :amod, :appos, :nn} *The AES11 standard published by the Audio Engineering Society provides a systematic approach to the synchronization of digital audio signals.* ***Entity** = AES11 ; **Type** = Standard*

We created a pipeline of SPARQL requests with a specific processing order as shown in Table 5.1. In fact, Pattern 1 has a higher priority than Pattern 2. For example, in the sentence *Sant'Elmo is the name of both a hill and a fortress in Naples, located near the Certosa di San Martino,* if the second pattern was processed before the first one, we would wrongly extract the type *name*. Similarly, Pattern 4 is the pattern with the lowest priority, which is executed only after all the other patterns are tested.

Each pattern is modeled using a single SPARQL request. The following is an example of the SPARQL implementation of Pattern 2:

```
SELECT ?typeLabel
WHERE {
       ?type :nsubj ?entity.
       ?type :cop ?cop.
       ?entity rdfs:label ?entityLabel.
       ?type rdfs:label ?typeLabel.
       FILTER(REGEX(?entityLabel, '^THE_LABEL_OF_THE_DBPEDIA_ENTITY', 'i')).
}
```

As this SPARQL request shows, we search for the type of an entity, where the entity's label is a perfect match with the input DBpedia entity's label. We first execute all the SPARQL patterns in this "full match" mode. In case all requests fail to return a type, we then look for occurrences of the same patterns using a partial match of the entity's label.

Once we find a candidate type, we create an OWL class representing this type. We remove all the accents and special characters and extract the lemma of the types in plural form. Overall, we adopted the singular as a convention for our entity types. For instance, in *Alvorninha is one of the sixteen civil parishes that make up the municipality of Caldas da Rainha, Portugal*, we extract the type *oke:Parish* from the string *parishes*. Finally, we create a *rdf:type* relation between the entity and the returned type.

5.4 Type alignment

In this paper, we refer to the namespaces *http://dbpedia.org/ontology/* and *http://dbpedia.org/page/* as *dbo* and *dbr* respectively. The ontologies *http://www.ontologydesignpatterns.org/ont/dul/DUL.owl* and *http://ontologydesign-patterns.org/ont/wikipedia/d0.owl* are represented by the prefixes *dul* and *d0* respectively. Besides, we use "Dolce" as a shortcut for "Dolce + DnS Ultralite".

Once the *natural language type* of a given DBpedia entity is identified, for instance [*dbr:Brian_Banner*, *oke:Villain*], where the first element represents the entity and the second element the natural language type, the second part of the OKE challenge task 2 is to align the identified type to a set of given types in the Dolce ontology[9]. The objective is to link the natural language type to a super-type in the ontology using an *rdfs:subClassOf* link. For instance, *dul:Person* would be a possible super-class for *oke:Villain*.

Our alignment strategy relies only on the DBpedia knowledge base and its links to external knowledge bases, when applicable, and exploits available mappings between DBpedia and Dolce, and Yago and Dolce. In fact, besides the OKE challenge in itself, one objective of this research is to determine whether the DBpedia knowledge base, one of the main hubs on the Linked Open Data cloud, is a suitable resource for the entity linking task. Thus our goal is to find a link, either directly or indirectly, between the *oke* type (e.g. *oke:Village*) returned by the first subtask and the DBpedia ontology and/or Yago and/or Dolce.

5.4.1 DBpedia Type Identification

Our global alignment strategy first queries the DBpedia ontology (*http://dbpedia.org/ontology/[Input Type]*). If the type is not found as a DBpedia class, we query DBpedia resources (*http://dbpedia.org/resources/[Input Type]*) and either find direct types or infer candidate types using several strategies. Our queries result in three possible outputs:

1. There is a *dbo* resource for the input type. In our dataset, this case occurred in 83 out of 198 cases (42%).
2. There is only a *dbr* resource for this type. In this case, we attempt to find a predicate *rdf:type* between the natural language type and some type that can be aligned with Dolce + DnS Ultralite, i.e. a type in the DBpedia ontology, Yago-Wordnet or DUL. In our dataset, this case occurred in 68% of the cases.

---

[9] https://github.com/anuzzolese/oke-challenge-2016#task-2

3. There is neither a *dbo* nor a *dbr* resource for this type (e.g. *oke:Villain*). In this case, we cannot infer any type and we rely solely on the entity page (dbr:Brian_Banner) to identify a potential type when possible. In our training data set, this case never occurred.

Next, we assign a score to our candidate types based on the number of instances available for these types. Finally, we return the Dolce + DnS Ultralite type that is equivalent or is a super-type of the chosen DBpedia type. The following sections describe the various implemented strategies for type alignment.

Method 1: Alignment Based on the DBpedia Ontology: The first method checks if there is full match between the natural language type and a class in the DBpedia ontology (e.g. for the input "*oke:Villain*", we look for the URI dbo:Villain). If such a class exists, we simply align this type with Dolce.

Method 2: AlignmentBbased on the type of Instances: In this step, the idea is to exploit the "informal" types available in the *dbr* namespace using the predicate *dbo:type*. For instance, even though *dbr:Village* is not defined as a class, we can find the triple *dbr:Bogoria,_Poland dbo:type dbr:Village*. Thus, given that *dbr:Bogoria,_Poland* is also of type *dbo:Place*, our general hypothesis is that we can consider *dbr:Village* to be a subclass of *dbo:Place*. To choose among all the candidates, we consider all the instances (using *dbo:type*) of *dbr:Village*, and assign a score to each of their types (available through *rdf:type*) depending on the number of times in which they appear in relation with the instances of *dbr:Village*.

Method 3: Alignment Based on the Entity Type: In this strategy, we exploit the information available in the DBpedia entity page itself. In fact, if the given natural language type does not have any DBpedia page, or if that page does not contain any information that could allow us to infer a valid type, we search for a direct *rdf:type* relation in the **entity** description. For instance, for the input [*dbr:Adalgis*, *oke:King*], our assumption is that the triple : *dbr:Adalgis rdf:type dul:NaturalPerson* implies *oke:King rdfs:subClassOf dul:NaturalPerson*. All the (rdf) types of the entity represent our candidate types with an initial score of 1.

Method 4: Alignment Based on Direct Types: Here we query the DBpedia resource corresponding to the natural language type (e.g *dbr:Club*) and find the triples of the form *dbr:Club rdf:type [Type]* and return *[Type]*. Like in Method 3, all the candidates returned by this method have an initial score of 1.

Method 5: Alignment Based on Categories: This strategy exploits Wikipedia categories represented by the *http://dbpedia.org/page/Category:* namespace (*dbc*). Categories are indicated in most pages using the predicate *dct:subject*. The idea here is to look at all the categories in which a given type is included (for instance *dbc:Administrative_divisions*, *dbc:Villages*, *etc.* for *dbr:Village*), and then find the type(s) of all the elements in each of these categories. In this example, the category *dbc:Villages* contains several villages (such as *dbr:Mallekan*) of type *dbo:Place*. *dbo:Place* is therefore a candidate type for *dbr:Village*. Like in previous methods, this approach returns many candidates. Each type is given a score equal to the number of triples in which it appears.

Method 6: Alignment Based on Predicates' Domain and Range: This method infers a type for an entity by examining the *rdfs:domain* and *rdfs:range* of predicates that are used in the description of the DBpedia page associated with the natural language type. For instance, the two triples:

dbr:Marko_Vovchok dbo:birthplace dbr:Village

dbo:birthPlace rdfs:range dbo:Place

allow us to infer *dbr:Village rdf:type dbo:Place* using the information available in the range of the predicate. In this approach, we only take into account the *dbo* predicates, as the *dbp* (http://dbpedia.org/property) predicates typically do not have any domain or range specified. Like in method 2, we give each inferred type a score equal to the number of triples in which the type is used.

5.4.2 Dolce+DUL Alignment

Following all our type identification methods, we obtain a set of candidate types with a score. Next, we rely on the alignment between the DBpedia ontology and Dolce + DnS Ultralite to replace each *dbo* type with their *dul/d0* counterpart. The same is done to replace *yago* types with *dul/d0* types. However, as the set of types used by the OKE challenge does not include all *dul* types, we modify this alignment in the following way: if a *dul* type is not included in the set of OKE challenge types, we replace it by its closest ancestor that is included in the set. For instance, *dul:SocialPerson* is not an element of the OKE challenge set, but its super class, *dul:Person*, is available. Therefore, if our alignment returns *dbo:Band rdfs:subClassOf dul:SocialPerson*, our final output is *dbo:Band rdfs:subClassOf dul:Person*. In our experiments on the OKE dataset, this strategy did not work

well only with the *dul* types *dul:Concept* and *dul:Agent*, which do not have any parent in the OKE challenge set.

Next, the obtained set of *dul* candidates is very often a set of classes that have some taxonomical link among themselves. Given that our objective is to find the most precise candidate, the score of each candidate is modified by adding the score of its ancestors among this set, thus effectively favoring classes that are deeper in the taxonomy. Finally, the chosen candidate is the *dul* type with the highest score.

Here is a full example of our process for method 2 (instances) with the input (*dbr:Calvarrasa_de_Abajo*, "*oke:Village*"). First, we retrieve all the URIs that appear in a triple of the form *[subject] dbo:type dbr:Village*. Then, we retrieve all the types (*rdf:type*) of URIs of the form: *[subject] rdf:type [dbo_type]*. Each of these types' score increases by 1 every time it appears. In this example, the final list contains 26 types, with the best (score-wise) being: *dbo:Place* (480), *dbo:Location* (480), *dbo:PopulatedPlace* (480), *dbo:Settlement* (480), *dbo:Village* (460), *yago:location* (436) and *yago:object* (436). After the DOLCE alignment, this list becomes *d0:Location* (1905), *dul:PhysicalObject* (437), *dul:Object* (436) and *dul:Region* (436). During this step, if several types are aligned to the same *dul/d0* type, their scores are combined.

Finally, we check if our candidates include *dul* types that are not available in the OKE challenge set, and replace them by their equivalent (if available) or closest ancestor type that is available in the OKE set. Here, *dul:Region* is replaced by *d0:Characteristic*. We end up with *d0:Location* (1905), *dul:PhysicalObject* (873), *dul:Object* (436) and *d0:Characteristic* (436). The type with the highest score is *d0:Location* (1905), therefore we return *oke:Village rdfs:subClassOf d0:Location.*

5.5 Evaluation

5.5.1 Type extraction evaluation

Our first evaluation calculates the precision and recall of the natural language type identification subtask. We consider a type as a true positive only when its lemmatized oke type is a perfect match

with at least one of the lemmatized oke types of the OKE gold standard[10]. Using this evaluation method on the 2016 train dataset, our precision and recall for the type extraction subtask is 87% as shown in Table 5.2, and 80% on the evaluation dataset as shown in Table 5.3. Table 5.5 presents the official evaluation on the 2016 train dataset using Gerbil[11] [60]. We can notice some decrease in performance using Gerbil (Table 5.2 and Table 5.5), some of which can be explained by the existence of OKE types in plural form in the gold standard (e.g. *oke:Awards* versus *oke:Award*). In fact, contrary to Table 5.5, the results in Table 5.2 take into account the lemmatization of both the natural language types and the gold standard types.

We performed an analysis of the unsuccessful sentences and identified few potential sources of errors. A good proportion of errors arise from grammatical ambiguities and incorrect syntactic analyses of sentences. This is the case for sentences like *Brad Sihvon was a Canadian film and television actor*, for which we find the type *oke:Film* instead of *oke:Actor*, due to an error in the parsing process. Similar errors can also occur, in some rare cases, for long sentences like *Bradycardia, also known as bradyarrhythmia, is a slow heart rate, namely, a resting heart rate of under 60 beats per minute (BPM) in adults* for which we extract the type *oke:Heart* instead of *oke:Rate*. In some cases, the errors are debatable. We list as examples the sentence *Gimli Glider is the nickname of an Air Canada aircraft that was involved in an unusual aviation incident*, for which we extract the type *oke:Aircraft* instead of *oke:Nickname*, or *Caatinga is a type of desert vegetation, and an ecoregion characterized by this vegetation in interior northeastern Brazil*, for which we find the type *oke:Vegetation* instead of *oke:TypeOfDesertVegetation*.

We also evaluated the precision and recall of our patterns separately. Given that the precision and recall are the same, Table 5.2 and 5.3 show the results for each pattern based on the 2016 OKE train and evaluation datasets.

Table 5.2 Statistics for the type extraction evaluation on the train dataset

Pattern	**(1)**	**(2)**	**(3)**	**(4)**	**Total**
Found Types	10	156	2	4	172

[10] https://github.com/anuzzolese/oke-challenge-2016/blob/master/GoldStandard_sampleData/task2/dataset_task_2.ttl

[11] http://gerbil.aksw.org/gerbil

Total Occurrences	14	173	2	9	198
Precision/Recall	71%	90%	100%	44%	87%

Table 5.3 Statistics for the type extraction evaluation on the evaluation dataset

Pattern	**(1)**	**(2)**	**(3)**	**(4)**	**Total**
Found Types	1	39	0	0	39
Total Occurrences	1	47	0	2	50
Precision/Recall	100%	82.98%	-	0%	80%

5.5.2 Type alignment evaluation

To assess the efficiency of each type alignment method, we compared the obtained types to those present in the gold standard. Table 5.4 shows the number of returned types, the number of correct types, as well as the precision, recall and F-measure for each method on the OKE challenge train dataset. Taken individually, most methods achieve limited or poor performance. However, we also implemented a pipeline strategy to combine these methods, thus increasing the recall of our approach. The pipeline is based on the most successful to the least successful strategies (in terms of precision) based on the results of individual methods. In this pipeline, a strategy is executed only if the previous one was unsuccessful in returning a type.

Table 5.4 Comparison of each method for type alignment on the OKE challenge train dataset

Method	Returned types	Correct types	Precision	Recall	F-measure
1: ontology	83	59	71%	30%	42%
2: instances	57	38	67%	19%	30%
3: entity	140	67	48%	34%	40%
4: direct type	17	6	35%	3%	6%
5: category	130	22	15%	10%	12%
6: predicates	89	11	12%	6%	8%
Pipeline 1 – 6	172	96	56%	48%	52%

5.5.3 Overall results

The overall results of the task of entity typing and alignment on the OKE training dataset using the evaluation framework Gerbil are shown in Table 5.5. These results rely on the pipeline strategy for the type alignment subtask.

We can notice a slight decrease in performance compared to our local evaluation on the 2016 train dataset.

Table 5.5 Overall precision, recall and F-measure computed using Gerbil on the 2016 train dataset

Task	Micro Precision	Micro Recall	Micro F-Measure	Macro Precision	Macro Recall	Macro F-Measure
Type extraction	82.32%	75.81%	78.93%	82.32%	78.96%	80.05%
Type alignment	49.63%	45.47%	47.45%	49.62%	45.42%	46.47%
Total (average)	65.97%	60.64%	63.19%	65.97%	62.19%	63.26%

The system was also tested on the 2016 test dataset; the result for this dataset are presented in Table 5.6.

Table 5.6 Overall precision, recall and F-measure computed using Gerbil on the 2016 test dataset

Task	Micro Precision	Micro Recall	Micro F-Measure	Macro Precision	Macro Recall	Macro F-Measure
Type extraction	81.63%	73.39%	77.29%	80.81%	76.60%	77.95%
Type alignment	46.46%	42.51%	44.40%	46.46%	42.51%	43.53%
Total (average)	64.05%	57.95%	60.85%	63.64%	59.55%	60.74%

For comparison purposes, we also report the results of two competing systems on the evaluation dataset in Table 5.7. These systems are Mannheim [25], a participant to the OKE 2016 challenge and CETUS [24], the baseline system and the winner of the OKE 2015 challenge. Mannheim uses taxonomical relation ("isa") extraction based on Hearst-like patterns in text to find the entity type, then chooses one of these isa relations and exploits a mapping between OntoWordnet, Wordnet and DOLCE to infer the super class in the OKE types set. CETUS uses pattern extraction to identify potential types, then creates a hierarchy between these types. Finally, it proposes two approaches to align the type with DOLCE: the first one is based on a mapping with Yago, and the second on an entity recognition tool (FOX).

Our system WestLab obtains satisfying results in terms of precision for the type extraction task; we obtain a Micro and Macro values of 86%, whereas the baseline CETUS obtains 68.75% (micro) and 72% (macro), and Mannheim obtains 77.27% (micro) and 63% (macro).

As for the recall, we obtain 86% (micro and macro), which is better than Mannheim's recall of 68% (micro and macro), but lower than CETUS which obtains 88% (micro and macro).

Our Micro and Macro F-Measures are both 86%. These results are higher than CETUS', which are 77.19% and 77.33% respectively, and Mannheim's, that obtains 72.34% and 64.67% respectively, for the type extraction.

Table 5.7 Overall precision, recall and F-measure for the two participating systems and the baseline CETUS on the test dataset

System	Task	Micro			Macro		
		Precision	Recall	F1	Precision	Recall	F1
CETUS	Extr.	68.75%	88.00%	77.19%	72.00%	88.00%	77.33%
	Align.	22.17%	24.47%	23.26%	22.17%	24.47%	19.89%
	Total	45.46%	56.24%	50.23%	47.08%	56.24%	48.61%
Mannheim	Extr.	77.27%	68.00%	72.34%	63.00%	68.00%	64.67%
	Align.	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	Total	38.64%	34.00%	36.17%	31.50%	34.00%	32.33%
WestLab	Extr.	86.00%	86.00%	86.00%	86.00%	86.00%	86.00%

	Align.	8.00%	6.67%	7.27%	8.00%	6.67%	7.00%
	Total	47.00%	46.33%	46.64%	47.00%	46.33%	46.5%

Thus, we can conclude that we outperform other systems when taking into account precision but we note that our recall is lower than the one obtained by the baseline CETUS. These results also show that our patterns do not always detect and extract the type of the entity, which is an indicator that the patterns set must be extended in our future work. However, our patterns rarely extract types that are false positives, which shows that they are well defined and accurate.

Concerning the type alignment, there have been some issues with the test dataset distributed by the OKE challenge organizers at the time of the evaluation, which have been corrected later. This explains the very low performance shown in Table 5.7 for the type alignment subtask. Given this modification, we are able to provide results only for our system and the CETUS baseline on the corrected evaluation dataset. At the time of this publication, we don't have the updated results for the Mannheim system. Table 5.8 provides our results on the updated dataset, compared with the baseline. Overall, we can notice a huge improvement on the corrected dataset. In fact, the WestLab system obtains an F-Measure of 44.4% (micro) and 43.53% (macro) for the type alignment subtask, whereas CETUS obtains 23.26% and 19.89%. These results constitute a considerable improvement for the type alignment task, even though they are still under the threshold of 50%.

For the overall results including both type extraction and alignment, we outperform all systems and obtain F-Measures of 60.85% (micro) and 60.74% (macro), whereas CETUS obtains F-Measures of 50.23% (micro) and 48.61% (macro). We cannot compare our system with Mannheim on the corrected test dataset, except on the type recognition subtask, for the reasons mentioned in the previous paragraph.

Table 5.8 Overall precision, recall and F-measure for the two participating systems and the baseline CETUS on the corrected test dataset

System	Task	Micro			Macro		
		Precision	Recall	F1	Precision	Recall	F1

CETUS	Extr.	68.75%	88.00%	77.19%	72.00%	88.00%	77.33%
	Align.	22.17%	24.47%	23.26%	22.17%	24.47%	19.89%
	Total	45.46%	56.24%	50.23%	47.08%	56.24%	48.61%
WestLab	Extr.	81.63%	73.39%	77.29%	80.81%	76.60%	77.95%
	Align.	46.46%	42.51%	44.40%	46.46%	42.51%	43.53%
	Total	64.05%	57.95%	60.85%	63.64%	59.55%	60.74%

5.6 Discussion

Type Extraction. One limitation of our approach for natural language type identification is the small number of implemented patterns, which does not guarantee to find an entity type. However, our proposal of SPARQL patterns, coupled with an RDF representation of definitions, represents an elegant and simple solution which facilitates the addition of new patterns. Another limitation comes from the fact that our system relies on a syntactic analysis. Thus, errors that occur in the parsing process also affect our system. However, according to our preliminary results, this approach displays a satisfactory precision and recall values compared to previous approaches in the OKE competition.

DBpedia for Type Alignment. Task alignment requires the discovery of *rdfs:subClassOf* links between natural language types and ontological classes. One of our research objectives was to assess how well a type alignment could be performed based on the structured knowledge available in the DBpedia ontology and resources. Some of our methods exploit the grey zone around the notion of subclass and instance in DBpedia. In fact, DBpedia resources (A-box) cannot be normally expected to use the *rdfs:subClassOf* predicate. However, some of the resources employ the predicate *dbo:type*. For example, *dbr:Bogoria,_Poland dbo:type dbr:Village*. Thus *dbr:Village* can be effectively considered as a *class* based on RDFS semantics. There were 57 (out of 198) similar cases in our train dataset. Based on this line of thought, if we found *dbr:Village rdf:type dbo:Place*, we inferred *dbr:Village rdfs:subClassOf dbo:Place.* These examples show that DBpedia resources (A-box) are also described using an informal or implicit schema. This further highlights the need of describing these resources in the ontology rather than in the knowledge base.

Due to the lack of directly exploitable type information in DBpedia, we relied on type inference methods (M2 - instances, M6 - predicates, M5 - categories) in few cases (27 out of 172 types are retrieved using these methods). More specifically, we employed these strategies when an input type does not have a *dbo* page or when its *dbr* page does not contain any *rdf:type* predicate. However, these methods often give poor results. Finally we did not process the disambiguation pages (e.g. *dbr:Motion*) that are sometimes returned by our methods. Altogether, our system failed to return any type in 14% of the cases. In this case, it returns *owl:Thing*.

Examples of Problematic Cases. Most of our errors boil down to two error sources: a) inaccurate, noisy, or plain false information and b) unavailable information in DBpedia. In the following, we give a few examples of problematic cases in some of the alignment methods.

M2 – instances: According to the gold standard, *dbr:Court* should be a *dul:Organization*. However, in DBpedia, *dbr:Court* instances, as depicted by the *rdf:type* predicate, are inaccurate (e.g. *dbr:Mansion_in_Grabowo_Krolewskie)* or refer to broken links. Our type alignment based on these links wrongly concludes that *dbr:Court* is a *d0:Location*.

M6 – predicates: *dbr:Season* should be a *dul:Situation*. However, in DBpedia, there is a confusion between a season (time of the year) and seasonal music (such as Christmas songs) which does not have a *dbr* resource. Therefore, the resource *dbr:Season* is used erroneously instead of the non-existing *dbr:Seasonal_Music* page. This leads to triples such as *dbr:Christmas _(Kenny_Rogers_album) dbo:genre dbr:Season.* Given that the predicate method exploits *dbo:genre rdfs:range dbo:Genre*, we erroneously conclude that a *dbr:Season* is a subclass of *dbo:Genre*.

M5 – categories: *dbr:Tournament* is part of only one category, *dbc:Tournament_systems*, containing pages such as *dbr:Round-robin_tournament* or *dbr:Double-elimination_tournament*. All of these resources have a type in Yago (*artifact*) that is aligned to *dul:PhysicalObject*, which makes us conclude that a *dbr:Tournament* is a *dul:PhysicalObject*. Here, the error is double: *dbr:Tournament* should not be in the category *dbc:Tournament_systems*, and the resources should not be typed as *yago:Artifact*.

In all the above examples, the correct answer is never present in our candidates list. This observation confirms that DBpedia resources are often poorly described [43]. Despite these

limitations, our pipeline, which is based on a set of methods ordered from the most trustworthy to the least one, obtains a micro precision of 49.6% on the training dataset and 46.5% on the test dataset, and micro recall of 45.5% on the training dataset and 42.5% on the test dataset, which we consider as reasonable given the complexity of the task.

Gold Standard. We had some issues when comparing our results with the gold standard. Quite often, our results could be considered as correct, but are different from the ones in the gold standard as they are based on the DBpedia ontology. For instance, we infer that a *oke:Meeting* is a subclass of *dul:Event* (*dul:Event* : "Any physical, social, or mental process, event, or state"), but the gold standard states that a *oke:Meeting* is a subclass of *dul:Activity*. Both answers could be acceptable. In the OKE train dataset, we identified 20 "borderline" cases out of 198 in the alignment subtask. In the natural language type extraction subtask, we identified some potentially questionable types in the gold standard of the form "Set_Of_X" or "Type_Of_X". For instance, in the sentence *Caatinga is a type of desert vegetation…* our position is that the type could be *oke:DesertVegetation* rather than *oke:TypeOfDesertVegetation*.

Future Work. For the type alignment sub-task, our next step will consider the problem of the disambiguation pages. Such pages represent a non-negligible portion of the data set (26%), and systematically constitute a source of errors. The objective is to choose the correct type among all the possible disambiguations. For instance, given the input [*dbr:Babylonia, oke:State*], the returned type *dbr:State* is a disambiguation page, linking to pages such as *dbr:Nation_state*, *dbr:State_(functional_analysis)* or *dbr:Chemical_state*.

5.7 Conclusion

This paper describes our approach for the extraction of entity types from text and the alignment of these types to the Dolce+DUL ontology. The patterns used to extract natural language types from textual definitions achieved high precision and recall values. As for the type alignment, the strength of our approach is based on the multiplicity of strategies which exploit both the DBpedia ontology and knowledge base and rely on DBpedia large coverage. Our experiments highlight the necessity of a better linkage between DBpedia resources and the DBpedia ontology and the need for restructuring some DBpedia resources as ontological classes.

5.8 Acknowledgement

This research has been funded by the NSERC Discovery Grant Program.

5.9 References

OKE 2015 Challenge Description. Last retrieved on March 20th 2016 from https://github.com/anuzzolese/oke-challenge/blob/master/participating%20systems/OKE2015_challengeDescription.pdf

Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investigationes 30(1), 3–26 (Jan 2007).

Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora (1992).

Min, B., Shi, S., Grishman, R., Lin, C.Y.: Ensemble semantics for large-scale unsupervised relation extraction. In: 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. pp. 1027–1037. Association for Computational Linguistics (2012).

Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL, pp. 1003–1011. ACL, Association for Computational Linguistics (2009).

Otero-Cerdeira, L., Rodriguez-Martinez, F.J., Gomez-Rodriguez, A.: Ontology matching: A literature review. Expert Systems with Applications 42(2), 949 – 971 (2015).

Bizer, C., Heath, T., Ayers, D., Raimond, Y.: Interlinking Open Data on the Web. Media 79(1), 31–35 (2007).

Giunchiglia, F., Shvaiko, P., Yatskevich, M.: Discovering missing background knowledge in ontology matching. In: Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence, pp. 382–386. IOS Press (2006).

Kachroudi, M., Moussa, E.B., Zghal, S., Ben, S.: Ldoa results for OAEI 2011. Ontology Matching p. 148 (2011).

Röder, M., Usbeck, R. and Ngonga Ngomo, A.: CETUS - A Baseline Approach to Type Extraction. In: OKE Challenge 2015 co-located with the 12th Extended Semantic Web Conference (ESWC) (2015).

Gao, J. and Mazumdar, S.: Exploiting Linked Open Data to Uncover Entity Types. In: OKE Challenge 2015 co-located with the 12th Extended Semantic Web Conference (ESWC) (2015).

Consoli, S. and Reforgiato, D.: Using FRED for Named Entity Resolution, Linking and Typing for Knowledge Base population. In: OKE Challenge 2015 co-located with the 12th Extended Semantic Web Conference (ESWC) (2015).

Bos, J.: Wide-Coverage Semantic Analysis with Boxer. In: Johan Bos and Rodolfo Delmonte, editors, Semantics in Text Processing, pages 277–286. College Publications (2008).

Ontology Design Patterns. (2014). Last retrieved on March 20th 2016 from http://ontologydesignpatterns.org/ont/dbpedia_2014_imports.owl

CETUS (2015). DolCE_YAGO_mapping. Last retrieved on March 20th 2016 from http://github.com/AKSW/Cetus/blob/master/DOLCE_YAGO_links.nt

Font, L., Zouaq, A., Gagnon, M.: Assessing the quality of domain concepts description in DBpedia. In: 11th International Conference on Signal-Image Technology and Internet-Based Systems, pp. 254-261, IEEE (2015).

De Marneffe, M-C, and Manning, C. D. The Stanford typed dependencies representation. Proc. of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation, ACL. (2008).

Usbeck, R., Röder, M., Ngonga Ngomo, A.-C., Baron, C., Both, A., Brümmer, M., Ceccarelli, D., Cornolti, M., Cherix, D., Eickmann, B., Ferragina, P., Lemke, C., Moro, A., Navigli, R., Piccinno, F., Rizzo, G., Sack, H., Speck, R., Troncy, R., Waitelonis, Jö. & Wesemann, L. GERBIL -- General Entity Annotation Benchmark Framework. *24th WWW conference*, (2015).

Faralli,S., Ponzetto, S. DWS at the 2016 Open Knowledge Extraction Challenge: A Hearst-like Pattern-Based Approach to Hypernym Extraction and Class Induction. In: OKE challenge 2016 co-located with the 13th Extended Semantic Web Conference (ESWC) (2016).

CHAPITRE 6 ARTICLE 3: AXIOPEDIA: ENRICHING DBPEDIA WITH OWL AXIOMS FROM WIKIPEDIA

Lara Haidar-Ahmad, Amal Zouaq, Michel Gagnon: AXIOpedia: Enriching DBpedia with OWL Axioms from Wikipedia. Soumis à Semantic Web Journal, le 4 avril 2017.

Abstract. The Semantic Web relies on the creation of rich knowledge bases which link data on the Web. Having access to such a knowledge base enables significant progress in difficult and challenging tasks such as semantic annotation and retrieval. DBpedia, the RDF representation of Wikipedia, is considered today as the central interlinking hub for the emerging Web of data. However, DBpedia still displays some substantial limitations such as the lack of class definitions and the lack of significant taxonomical links. The objective of this work is to enrich DBpedia with OWL-defined classes and taxonomical links using open information extraction from Wikipedia. We propose a pattern-based approach that relies on SPARQL to automatically extract axioms from Wikipedia definitions. We run the system on 12,901,822 Wikipedia pages (including disambiguation pages). The resulting knowledge base, AXIOpedia benefits from a rich and consistent ontology with complex axioms, rdf:subclassOf and rdf:type relations.

6.1 Introduction

Nowadays, the Semantic Web, and more specifically linked data, relies mostly on lightweight ontologies. However, the need for richer and more complete knowledge bases has emerged for solving some of the most difficult and challenging tasks of the Semantic Web [1] like powerful querying, reasoning and inference, and semantic annotation.

In this context, the DBpedia project started as a community effort to extract information from Wikipedia, and represent them in an RDF knowledge base. Wikipedia pages consist mainly of natural language text, but also contain semi-structured information in the form of wiki markups, such as infoboxes, categories, images, geo-coordinates, and links to external web pages. As of

November 2016, the English Wikipedia contains 5,290,915 articles[12], and the DBpedia dataset contains 4,305,028 instances[13] related to each other with RDF triples, most of which are extracted from (semi)structured information from Wikipedia, neglecting most of the information that can be extracted from natural language text.

DBpedia has become a central interlinking hub for the emerging Web of data. It relies on an ontology that mainly consists of named classes, taxonomical links and object properties, but it lacks defined classes. Defined classes are useful for inferring implicit knowledge. For example in DBpedia *dbr:Tahiti* is typed as a *dbo:Island*, which is not defined by an axiom. However, the term *country* is defined in Wikipedia as *An island or isle is any piece of sub-continental land that is surrounded by water*. Thus, by having the axiom $Island \equiv Isle \equiv pieaceOf.SubContinentalLand \cap \exists surroundedBy.Water$, we could have inferred that *dbr:Tahiti* is a piece of sub-continental land, and that it is surrounded by water.

Moreover, it has been shown that DBpedia representation of domain knowledge in particular is very limited [63] and that the DBpedia ontology could be more fine-grained. Thus, enriching DBpedia with new classes, instances, taxonomical and conceptual links from Wikipedia content is still a timely topic. In this work, our objective is two-fold:

- Extract AXIOPedia, an axiomatic ontology and knowledge base from Wikipedia definitions, by learning axioms defining classes and taxonomical links;
- Offer a REST-based Web service[14] to the research community for the automatic extraction of OWL axioms from Wikipedia.

To our knowledge, and despite several efforts to mine knowledge from Wikipedia [64], none of the existing works have completely tackled the above stated limitations. For example, none of the existing works try to extend DBpedia with a more expressive ontology. Additionally, one of the notable aspects of this work is that we rely on SPARQL pattern-matching capabilities to define our

---

[12] https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia

[13] http://wiki.dbpedia.org/services-resources/datasets/dataset-2015-10/dataset-2015-10-statistics

[14] www.westlab.polymtl.ca/AXIOpediaWebApp/Home

patterns and construct axioms [4]. SPARQL and RDF are standards of the semantic Web and thus it is interesting to use these models for pattern searching. Thus, we offer a solution that exploits Semantic Web languages and models to develop a Semantic Web knowledge base.

Finally, as this will be further explained, a part of this work is an extension of the second task of the 2016 OKE challenge[15] at the Extended Semantic Web Conference. In this challenge [66], we obtained the best performance for entity typing from natural language definitions. This approach is extended in the REST service presented here to type instances and populate classes in AXIOpedia with these instances.

This paper is structured as follows; Section 6.2 presents the state-of-the-art and related work. Section 6.3 describes an overview of our work and our main motivation. Section 6.4 explains in detail our approach and the system's architecture. Section 6.5 is dedicated to our evaluation and results. Finally, Section 6.6 concludes and discusses our work and future improvements.

6.2 Related Work

DBpedia is one of the richest knowledge bases on the linked open data cloud: [6] evaluates that resources on DBpedia have on average 47.19 triples, which according to [6], is higher than most other linked open datasets. The study also reveals that DBpedia contains errors and inconsistencies in the data, estimating that a resource has on average 5.69 problems. Also, DBpedia contains mainly data extracted from structured information [7], neglecting the information that can be found in natural language. Many studies have been dedicated to enrich or maintain DBpedia [69, 70, 71, 72]. For example, [70] looks for inconsistencies in DBpedia to maintain a good quality of results, while [71] is a system to keep DBpedia up-to-date with the new Wikipedia releases. Nonetheless, regardless of the efforts to enrich DBpedia and minimize its flaws, DBpedia has still a lightweight ontology which lacks essential axioms.

Wikipedia being the largest collaborative encyclopedia on the web, many researchers have worked on extracting automatically information from its content, and adding this data to DBpedia and/or other knowledge bases. [12] explains the important role of Wikipedia in data mining and its

[15] https://github.com/anuzzolese/oke-challenge-2016/blob/master/README.md

convenience for tasks related to automatic data extraction. Indeed, Wikipedia's characteristics, such as the link structure and anchor texts, makes it a proper candidate for data extraction. [13] outlines four categories of state-of-the-art methods for mining knowledge from Wikipedia: natural language processing on Wikipedia, information retrieval, information extraction, and ontology building. Indeed, building a rich ontology is a complex task which requires a thorough study in those four fields, but also requires a wide variety of knowledge; as explained in [75], the ontology layer cake, displayed in Figure 6.1, exposes the 6 layers on which an ontology relies. The first layer is the layer containing terms, some of which represent the same concepts and are defined as synonyms in the second layer. Terms represent concepts, defined in layer 3, that are organized in a taxonomy, in layer 4, and are connected with non-hierarchical relations, in layer 5. In addition, rules and axioms are defined in layer 6, to document the ontology and to allow the inference of new knowledge. The 6 layers (terms, synonyms, concepts, taxonomy, non-hierarchical relations, and axioms and rules) combined together represent the set of required knowledge in an ontology.

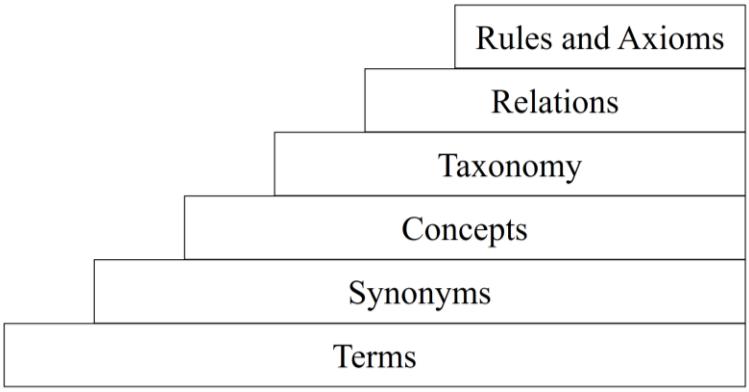

Figure 6.1 Ontology Layer Cake

Each layer of the ontology layer cake is addressed by various works. However, to our knowledge, there isn't any work that engages in the creation of an ontology which considers thoroughly all of these layers, especially in layer 6 (Rules and axioms). Hereafter, we will focus on layers that require the extraction of predicates and complex axioms.

In terms of the extraction of taxonomical and *instance-of* relations from Wikipedia, the goal is the identification of *is a* relations. Many methods rely on Wikipedia's structured information to identify pages' types, like the infoboxes and the categorization information. [76] relies on Wikipedia's system of categories to generate taxonomies by creating *rdf:type* and *rdfs:SubClassOf* relations between the pages and their respective categories. In the same vein, the objective of 2016

OKE challenge[16] (task 2), for which our system was the winning approach, aims at creating *rdf:type* statements between a given entity and its type from natural language definitions. A second aspect of the task is to align the extracted type to a set of given types from the DOLCE+DnS Ultra Lite ontology. Other participants to this challenge in the 2015 and 2016 editions include CETUS [77], OAK@Sheffield [78], FRED [79], and Faralli and Ponzetto's system [80]. CETUS extracts entity types from text by relying on rules on parts-of-speech (POS). OAK uses machine learning methods and POS patterns to recognize the sentences' portions that express the entity types. FRED is based on the system Boxer [81] and Discourse Representation Theory, and thus relies on a complex architecture for ontology extraction that is not limited to type extraction. Finally, Faralli and Ponzetto use an approach based on Hearst-like patterns [82]. Aside from the system participating in the OKE challenge, LHD 2.0 [83] uses an interesting approach for type extraction; while most approaches choose between the two techniques, LHD 2.0 combines both lexico-syntactic pattern analysis with supervised classification. By doing so, they are able to type 70% of the untyped instances in DBpedia. They map the types with DBpedia URIs using perfect string matching when possible, and otherwise using a more complex module based on statistical methods. Our approach for this task [66] represents a middle ground between patterns based on a superficial representation of sentences (usually parts of speech) and approaches such as FRED, which depend on complex first-order logic and frame semantics.

There are several works on the extraction of conceptual relations (object properties) from Wikipedia content. For example, [84] relies on the hierarchy of categories of Wikipedia and the structure of the name of these categories, to extract non-taxonomical relations. Other approaches focus on extracting information from text instead of infoboxes and structured information. These approaches [21, 24] require to process the language and resort to methods like pattern matching, parsing, and statistical learning. WOE [85], that extend from the work done by [86], is an example of a statistical learning method. WOE is based on the open extraction paradigm without direct supervision (like annotated training examples or hand-crafted patterns). To do so, they resort to the automatic construction of training datasets by heuristically matching Wikipedia infobox values to

[16] https://github.com/anuzzolese/oke-challenge-2016/blob/master/README.md

their corresponding text, and by using POS or parsing dependencies for the detection of patterns in the detected text.

There are very few works resulting in an ontology with well-defined axioms and rules. The available state of the art can be decomposed into pattern-based approaches and statistical schema learning approaches [87, 88, 62]. [1] rely on instances in an ontology and recurring RDF triples between instances to learn possible axioms for the classes of these instances. While this approach can infer some interesting information, it only relies on information already existing in DBpedia and does not generate complete and complex axioms but rather parts of axioms. It can also lead to potential conceptual flaws when a rule is recurrent among some instances and is thus falsely interpreted as a rule defining the class; for example, the rule $Song \sqsubseteq \exists album.MusicalWork$ will be extracted, since the vast majority of songs in Wikipedia are part of an album, but counting this as a rule is incorrect since a song is not necessarily part of an album. Other approaches to achieve this goal are pattern-based approaches, LExO [87] is a tool that used an approach similar to ours to extracts class axioms from natural language definitions by relying on the grammatical structure of sentences. Our approach for this task is based on patterns and grammatical dependencies between words. Our system does not only generate an axiom, but it also creates URIs for the classes and resources and links them with taxonomical relations, properties and restrictions.

Our work relies on the extraction of information from natural language from Wikipedia to enrich DBpedia and extends it towards a rich and expressive ontology, by enriching the layer of taxonomical relations and the layer of axioms. More specifically, we focus on the identification of the nature of entities (classes or instances), the extraction of necessary (and sufficient) axioms defining classes –which require both named classes, properties and logical operators- and the typing of instances. We also deal with the synonyms' layer by carrying out the disambiguation of the created concepts. In the following section, we highlight the interest of our approach.

6.3 Overview and Motivation of this Work

As of today, Wikipedia contains 5M pages. Because our aim is to extract both an ontology and a knowledge base, it is important to be able to distinguish two types of Wikipedia pages: pages representing classes and pages representing instances. Classes model concepts of the world, such

as *Country*, *Person* and *Movie*, whereas instances represent unique entities in the world, like *Canada*, *Obama*, or *The Godfather*.

In an ontology, classes and instances differ in the nature of links they have with other elements in the ontology; the triples between two classes and the triples between an instance and a class are semantically different. For example, in a taxonomy, a class would be linked by a *rdfs:subClassOf* relation to another class (for example *Composer rdfs:subClassOf Person*) whereas an instance would have a relation *rdf:type* with a class (for example *Beethoven rdf:type Composer*).

To illustrate the information needed for a class, we will use, as an example, the class *Poet*. The Wikipedia page of *Poet* defines it as *a person who writes poetry*. The corresponding DBpedia page contains triples using the following properties:

- *dbo:abstract* and *rdfs:comment* linking it to a textual definition,
- *dct:subject* linking it to subjects like *dbc:Arts_occupations*, *dbc:Spoken_words*, and *dbc:Poets,*
- *owl:sameAs* linking it to pages from other knowledge bases and pages in other languages,
- and many other properties linking instances to this class, like *dbr:Poet dbo:field dbr: Christopher_C._Bell*, and *dbr:Poet dbo:occupation dbr:Jules_Verne.*

Most of this information is correct but not valuable for the class definition. Defining *Poet* in this page with the axiom $Poet \equiv Person \cap \exists writes.Poetry$ in the ontology could help infer information about instances of this class. In this specific case, if an instance is a *Poet*, we can infer that this instance is also a *Person* and that this instance *writes Poetry*. Conversely, if we learn that someone wrote some poetry, we could automatically infer that this person is a poet.

For the instances, the only information that we are interested in are their type and potentially, their relations with other instances through OWL object properties. Given that our purpose is to extract defined classes, we can infer implicit knowledge about instances of those classes. We will illustrate this with the instance *Charles Baudelaire*. The Wikipedia page of *Charles Baudelaire* defines him as *Charles Pierre Baudelaire was a French poet who also produced notable work as an essayist, art critic, and pioneering translator of Edgar Allan Poe*.

The corresponding DBpedia page contains information about this instance like the birth name, birth date, occupation, type, etc. The information our system extracts is *CharlesBaudelaire rdf:type*

*FrenchPoet (*where *FrenchPoet rdfs:subClassOf Poet), CharlesBaudelaire rdf:type Essayist, CharlesBaudelaire rdf:type ArtCritic(*where *ArtCritic rdfs:subClassOf Critic), CharlesBaudelaire rdf:type Translator*. However, since *Poet* is defined above, we also indirectly have new information about *CharlesBaudelaire*.

6.4 Methodology and General Architecture

Our work relies on a pattern-based approach; by manually analyzing Wikipedia definitions, we detected recurrent syntactic structures indicative of axioms. Each pattern is mapped to a *construct* query that builds an OWL axiom in case of classes or a type statement in case of instances. In this section, we describe the overall architecture of our framework (Figure 6.2). The system takes Wikipedia definitions as input and distinguishes between instances and classes in the first step (Section 6.4.1). Then it generates an RDF graph representing the sentence and its grammatical structure according to a model described in section 6.4.2. This graph is then used to search for occurrences of syntactic patterns using SPARQL queries. SPARQL queries are also used to construct the mapped axioms or types (Section 6.4.3, 6.4.4 and 6.4.5). Finally, in section 6.4.6, we explain the disambiguation step that links our output with DBpedia.

6.4.1 Identification of Instances and Classes in Wikipedia

Determining accurately if a page represents an instance or a class is a complex task. The purpose of our work being mostly focused on the extraction of a knowledge base, this step represents only an elementary filtering of the classes and instances and more complex techniques for accurate detection are left for future work. In this paper, the system relies on several heuristics, some of which are similar to [89].

First, if there are any non-alphabetic characters (except spaces) in the title of the page, it is automatically considered as an instance, since titles like “Apollo 11” (numerical characters), “Who Wants to Be a Millionaire?” (punctuation), “Critics’ Choice Movie Awards” (diacritical marks) are likely to be names of instances. Second, we rely on the information already in DBpedia; if the corresponding DBpedia page is the *rdf:type* of another page, we conclude that we are dealing with a class. Third, if this information is missing or does not exist, we use Stanford NER [31] to determine if the title contains a named entity; if so, we consider the page as an instance. Fourth, we resort to the part-of-speech of the Wikipedia title extracted using Stanford parser [32]; in the

case of classes, the title should contain at least one noun and can only contain nouns and adjectives. Indeed, titles containing determinants, pronoun, verbs, or any part-of-speech other than nouns and adjectives, are most likely to represent instances, like “The Godfather” (determinant), “Love of my Life” (pronoun), “Somebody To Love” (verb). In our last heuristic, we verify if the DBpedia page has an *rdf:type statement*. If so it is considered as an instance and otherwise as a class.

As shown in our evaluation results (section 6.5.1), the use of all these methods together allows us to have a proper filtering of classes and instances for our next steps.

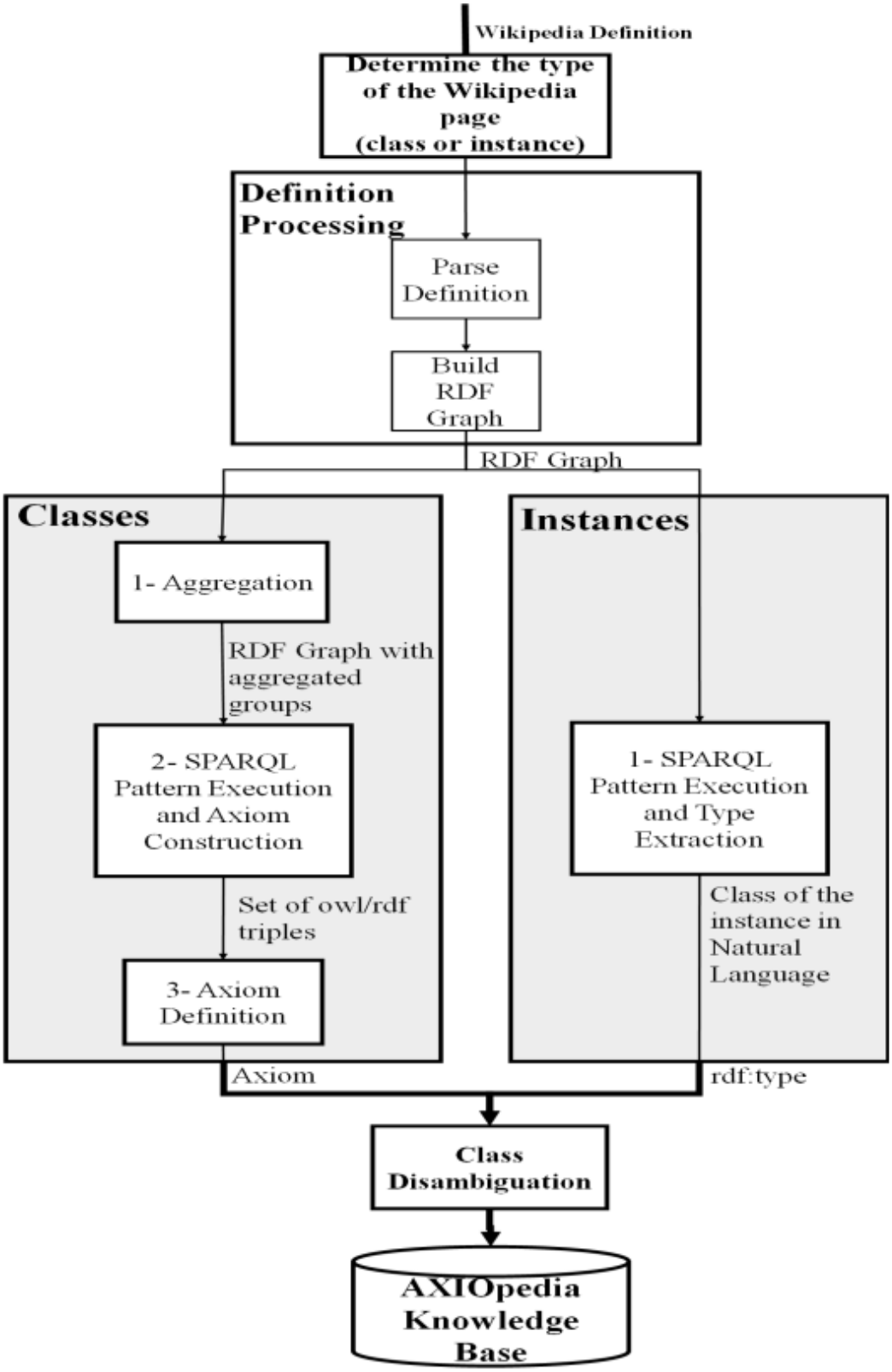

Figure 6.2 General Architecture

6.4.2 Processing the Textual Definitions

As mentioned above, we rely on a pattern-based approach to detect syntactic constructs. Given a definition from Wikipedia, we process the sentence and extract its grammatical analysis using the Stanford parser [32], and first construct an RDF graph representation. This graph is to make the subsequent step of pattern matching using SPARQL requests easier.

For more clarity, we will illustrate the process using, as an example, the definition of the class *Natural Satellite*[17]: *A natural satellite or moon is an astronomical object that orbits a planet or minor planet*. The RDF graph is illustrated in Figure 6.3. In the constructed RDF graph, we represent every word of the sentence with a node. Every node is related to its label (*rdfs:label*), its part-of-speech (*axio:pos*), its position in the sentence (*:position*), and its grammatical relations with other words of the sentence based on the output of the Stanford parser [32]. For example, the word *Satellite* in the sentence is represented by the node: Satellite-*3* and has the links *rdfs:label "Satellite"*, *axio:pos NN*, since it is a noun, *:position "3"*, since it is the third word in the sentence. It is also present in the following triples: Satellite-*3 :amod :Natural-2*, since *Natural* is a modifier of *Satellite* in the sentence, and *:Object-9 :nsubj :Satellite-3* since it is the subject of the sentence, etc.

[17] https://en.wikipedia.org/wiki/Natural_satellite

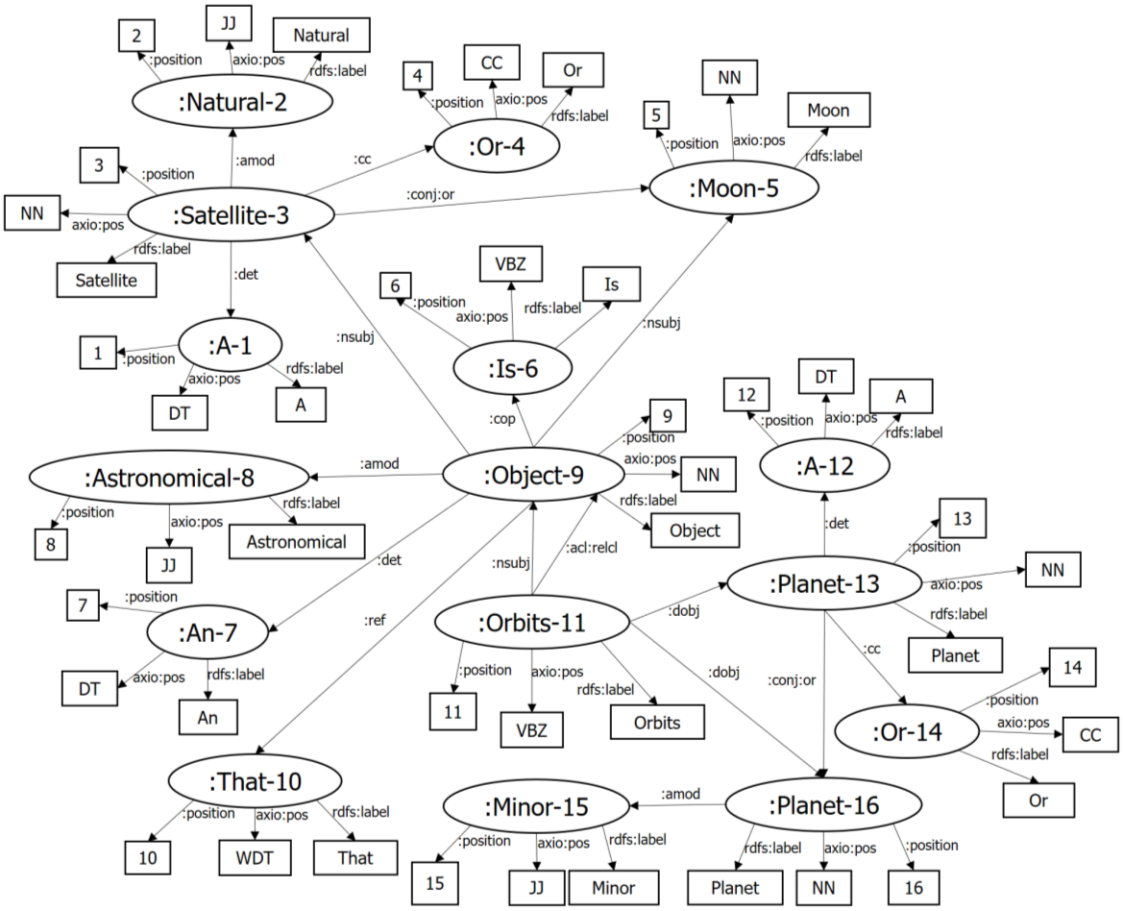

Figure 6.3 RDF Graph of the definition of Natural Satellite

Next, we run a pipeline of SPARQL requests on the obtained RDF graphs. Section 6.4.3 and 6.4.4 describe this pipeline respectively for classes and for instances.

6.4.3 Processing the RDF Graph to Extract Axioms for Classes

To extract axioms from the RDF graph, we go through 3 main steps. First, we start with the aggregation of nominal and verbal groups. Then we execute SPARQL requests to detect the occurrence of specific syntactic patterns in the RDF graph and construct partial axioms that correspond to these patterns. Finally, we combine partial axioms into one complex and complete axiom. These steps are explained in detail using the example of *Natural Satellite* given above.

6.4.3.1 Aggregation

In this step, we identify nominal and verbal groups and consider modifiers and conjunctions to aggregate these groups. We go through three sub steps to complete the aggregation process. First, we a) detect modifier nodes in the RDF graph; then we b) detect nodes that must be grouped into union and intersection lists and c) aggregate the groups iteratively.

a) **Detection of modifier nodes:** Words' modifiers like adjectives are aggregated to their head word. We represent these dependencies by creating triples *:node :dependency :modifier_node*, where each node points to its dependent nodes. The triples below show the triples added to the graph at the end of this step for our example:

:Sattelite-3 :dependency :Natural-2 .
:Object-9 :dependency :Astronomical-8 .
:Planet-16 :dependency :Minor-15 .

b) **Detection of union and intersection:** In some cases, the sentence describes several elements that should be grouped into union or intersection lists. In these cases, the dependency triples returned by the Stanford parser may not always be completely correct. In the above example, *:Planet-13* and *:Planet-16* are the *:dobj* of *:Orbits-11*, which means that two occurrences of "planet" are the objects of the verb *orbits* in *an object that orbits a planet or minor planet*. However, the object should actually be the whole nominal group *a planet or minor planet*. This is the reason why we build a list containing the two elements.

Therefore, in this step, we search for the coordinating conjunction *or* and *and*, and create respectively union or intersection lists containing the coordinated terms of the sentence. Note that in this step we do not consider the cases where the coordinator *or* is used to coordinate words that are considered as subjects, like in "*A natural satellite or moon is …*"; indeed, in these cases, we are dealing with an equivalence relation rather than a union (see Section 6.4.3.2).

Then, we create triples involving the node, its dependency relation, the type of the dependency (union or intersection), and the list. The output of this step is shown below:

These triples:

:Orbits-11 :dobj :Planet-13;
:dobj :Planet-16.

are replaced by:

:Orbits-11 :dobj
[
:dependency (:Planet-13, :Planet-16);
rdf:type axio:Union
].

c) **Axiom creation from aggregated groups:** In this step we detect the groups that have dependencies and aggregate them. The aggregation differs if the node represents a modifier or a list of coordinated elements. In the case of a simple node, we simply merge the labels of the nodes as shown:

These triples:

:Satellite-3 :dependency :Natural-2 .
:Object-9 :dependency :Astronomical-8 .
:Planet-16 :dependency :Minor-15 .

are translated into these taxonomical triples:

:NaturalSatellite owl:subClassOf :Satellite .
:AstronomicalObject owl:subClassOf :Object .
:MinorPlanet owl:subClassOf :Planet .

and we update the following triples in the RDF graph:

:Satellite-3 rdfs:label "Natural Satellite"
:Object-9 rdfs:label "Astronomical Object"
:Planet-16 rdfs:label "Minor Planet"

In the case of lists of elements, we create a complex class description. For instance,

These triples:

:Orbits-11 :dobj

[

:dependency (:Planet-13, :Planet-16)

].

are translated into the following triples:

:*Orbits-11 :dobj Planet_Or_MinorPlanet* .

Planet_Or_MinorPlanet owl:equivalentClass

[owl:unionOf (:Planet :MinorPlanet)] .

6.4.3.2 Pattern Execution and Axiom Extraction

The creation of axioms relies on recurring syntactic structures with their corresponding SPARQL patterns, which are mapped to complex class axioms using SPARQL CONSTRUCT. The list of patterns was manually established by analyzing a random set of Wikipedia definitions.

Table 1 shows the first four patterns that are executed on the RDF graph. Overall, we modeled 32 patterns. Each pattern is represented by a single SPARQL query that looks for the pattern and constructs the corresponding axiom. Pattern (3) is special, since it can be applied recursively on its element B. For instance, its application to *A landform is a natural feature of the solid surface of the Earth* would result into *naturalFeatureOf.(solidSurfaceOf.Earth).*

Table 6.1 Excerpt of the Patterns and their Mapped Axioms. The Complete list is presented in Table 6.13 (Annex I).

Patterns	**Corresponding Axioms**
(1) :neg A B *In Foreign exchange market, synthetic currency pair or synthetic cross currency pair is an artificial currency pair which generally is* ***not available*** *in market but one needs to trade across those pairs.*	$A \equiv \neg B$ $NotAvailable \equiv \neg Available$

(2) :cc :cc:or OR A B X :nsubj *A **natural satellite or moon** is an astronomical object.*	A ≡ B *NaturalSatellite ≡Moon*
(3) :nmod:of A B *Vehicles are non-living **means of transportation**.*	AOfB ≡ ∃AOf.B *NonLivingMeansOfTransportation ≡ ∃NonLivingMeansOf.Transportation*
(4) :nsubj :cop A B C ***A computer is a device** that can be* instructed *to carry out an arbitrary set of* arithmetic *or* logical *operations automatically.*	A ⊆ B *Computer ⊆ Device*
(11) :nsubj :cop :dobj A B C D E :acl:relcl :nsubj/:nsubjpass *Theatre or theater is a collaborative form of fine art that **uses live performers**, to present the experience of a real or imagined event before an audience in a specific place, often a stage.*	A ⊆ ∃D.E *Theatre ⊆ ∃uses.LivePerformers*

Figure 6.4 shows an example of a pattern (pattern (4) in table 6.1) implemented as a SPARQL request. This SPARQL request looks for the pattern's occurrences in the RDF graph and constructs the associated axiom.

```
CONSTRUCT
{
        ?subj a owl:Class.
        ?comp a owl:Class.
        ?subj rdfs:subClassOf ?comp.
}
WHERE
{
        ?comp :nsubj ?subj.
        ?comp :cop ?cop.
}
```

Figure 6.4 SPARQL implementation of Pattern (4)

The complete list of patterns is in Table 6.13 (Annex I). We also include Hearst Patterns [82] in Table 6.15 (Annex I), which we will further explain in Section 6.4.5.

The processing of the example *A natural satellite or moon is an astronomical object that orbits a planet or minor planet* involves 3 patterns shown in Table 6.13 (Annex I), with the following results:

The following triples in the graph:

:NaturalSatellite

:cc:or :Or ;

:conj:or :Moon.

Will result in the creation of the following axiom (one occurrence of pattern 2):

:NaturalSatellite ≡ *:Moon*

The following triples in the graph:

:AstronomicalObject

:nsubj :NaturalSatellite ;

:nsubj :Moon;

:cop :Is.

Will result in the creation of the following axioms (two occurrences of pattern 4):

$:NaturalSatellite \subseteq :AstronomicalObject$

$:Moon \subseteq :AstronomicalObject$

Finally, the following triples in the graph:

:AstronomicalObject

:nsubj :NaturalSattellite;

:nsubj :Moon;

:cop :Is.

:Orbits

:acl:relc AstronomicalObject;

:dobj Planet_Or_MinorPlanet .

:Planet_Or_MinorPlanet

owl:equivalentClass [owl:unionOf (:Planet :MinorPlanet)] .

Will result in the creation of the following axiom (two occurrences of pattern 11):

$NaturalSatellite \subseteq \exists orbits.(Planet \cup MinorPlanet)$

$Moon \subseteq \exists orbits.(Planet \cup MinorPlanet)$

6.4.3.3 Axiom Assembly

The last step simply consists of assembling the partial axioms into one complete axiom. If a class is a subclass of multiple classes or restrictions, we add them to an intersection list and create a subclass link between the class and this list instead of each one separately. For example, the previous steps built some of these axioms:

$NaturalSatellite \subseteq Satellite$

$NaturalSatellite \subseteq AstronomicalObject$

$NaturalSatellite \subseteq orbits.(Planet \cup MinorPlanet)$

This step assembles these axioms and generates:

$$NaturalSatellite \subseteq (Satellite \cap AstronomicalObject \cap \exists orbits.(Planet \cup MinorPlanet))$$

This axiom is the main axiom constructed for the definition of *Natural Satellite* in addition to taxonomical relations and relations like *Moon* ≡ *NaturalSatellite*, *AstronomicalObject* ⊆ *Object* and *MinorPlanet* ⊆ *Planet*.

6.4.4 Processing the RDF Graph to Extract rdf:type Triples for Instances

The process of type extraction for instances is similar to the process of axiom generation for classes. Similarly, we rely on the RDF graph built in 6.4.2 that represents the sentence and its grammatical relations and part of speech, and we search for syntactic patterns occurrences in the graph using SPARQL requests. However the built graph is modified in case we are dealing with an instance, and we consider the whole instance name as a single node; the reason is that, unlike classes' names, which usually consist of a single noun or a noun and an adjective, instances' names can be more complex, and can sometimes even be complete sentences, like for example the instance *All's Well That Ends Well* that represent the name of a play, which is defined in Wikipedia as *All's Well That Ends Well is a play by William Shakespeare*. While parsing sentences containing complex instance names, the Stanford parser considers the title's grammatical dependencies in the whole sentence's grammatical analysis, and the parsing is inaccurate. In order to solve this problem, we detect the Wikipedia page title in the sentence and aggregate it if it is a multi-words title. For example, in the sentence *All's Well That Ends Well is a play by William Shakespeare*, we identify *All's Well That Ends Well (*the Wikipedia title) and modify the sentence to obtain *AllsWellThatEndsWell is a play by William Shakespeare*. By aggregating the whole title, it is considered as a single word in the sentence and can then be correctly interpreted. We then parse the sentence and construct the RDF graph representing the dependency structure of the definition.

The rest of the method used for the extraction of instances' types is similar to the one explained above for processing classes but includes specific SPARQL queries relating an instance to its type. These patterns are presented in Table 6.2, and are executed in addition to the Hearst patterns (Section 6.4.5). Moreover, we make sure that the extracted type is a noun. Once the instance's class is detected, we extract the lemma, as we adopted the singular as a convention for our entity types. Then we aggregate the type with its modifiers and create a *rdfs:subClassOf* relation. For example, for *Charles Pierre Baudelaire was a French poet* we extract *FrenchPoet,* which will be the type of Baudelaire and will be a subclass of Poet. Finally we build an *rdf:type* between the instance and the class.

Table 6.2 Patterns for detecting Instances' Types

Patterns
(1) :nsubj :cop Entity X Z Type :nmod:of Where X = { "name", "nickname", "surname", "alias", "one" } *Sant'Elmo is the name of both a hill and a fortress in Naples, located near the Certosa di San Martino.* ***Entity** = Sant'Elmo ; **Type** = Hill, Fortress*
(2) :nsubj :cop Entity Type Z *El Oso, released in 1998, is an album by the New York City band Soul Coughing.* ***Entity** = El Oso ; **Type** = Album*
(3) :nsubjpass :auxpass Entity V Z Type :nmod:as *Bromius in ancient Greece was used as an epithet of Dionysus/Bacchus.* ***Entity** = Bromius ; **Type** = Epithet*
(4) x Entity Type Where x = { :compound, :amod, :appos, :nn} *The AES11 standard published by the Audio Engineering Society provides a systematic approach to the synchronization of digital audio signals.* ***Entity** = AES11 ; **Type** = Standard*

6.4.5 Hearst Patterns

Hearst Patterns [82] (see Table 6.15 in Annex I) are used for classes and instances. These patterns detect *is a* relations between two classes (translated as *rdfs:suclassOf*) or a class and an instance (translated as *rdf:type* links). For example, the sentence *Planets such as Saturn* illustrates a relation between the class *Planet* and the instance *Saturn*, and should lead to the creation of the triple *:Saturn rdf:type :Planet*, whereas the sentence *Astronomical Objects such as planets* illustrates a relation between the class *AstronomicalObject* and the class *Planet*, and should lead to the creation of the triple *:AstronomicalObject rdfs:subClassOf :Planet*.

If we are dealing with an instance, we extract an *rdf:type* triple using a single SPARQL query that creates a *rdf:type* in case the pattern is found. However, in case we are building an axiom, we process the whole sentence, and when a similar pattern is found in the sentence, we do not know if we should build a *rdf:type* relation or a *rdfs:subClassOf* relation. For this purpose, we process Hearst patterns in two steps. First we detect the occurrences of these patterns in the RDF graph with a SPARQL request, and construct an *:isA* relation. Then, for each *:isA* relation, we verify if its two arguments are both classes or a class and an instance, and create the associated relation accordingly. To specify if we are dealing with a class or an instance, we verify the label of the concerned node and check whether it has a Wikipedia page, and if so, we use the same approach explained in Section 6.3. If none of our heuristics are applicable, we suppose that we are dealing with an instance.

6.4.6 Axiom and Type Disambiguation

At this point, we have an axiom with textual entities in case of a class, or the extracted textual type in case of an instance. However, we need the URI of the corresponding DBpedia resources in order to link our output and integrate it to DBpedia. To do so, we resort to a disambiguation step.

For the example *A natural satellite or moon is an astronomical object that orbits a planet or minor planet*, after extracting the axiom $NaturalSatellite \subseteq (Satellite \cap AstronomicalObject \cap \exists orbits.(Planet \cup MinorPlanet))$, we must disambiguate every entity referred in this axiom and in the taxonomical relations, by linking them to their corresponding DBpedia URIs : *http://dbpedia.org/resource/Natural_satellite*, *http://dbpedia.org/ontology/Satellite*, *http://dbpedia.org/resource/Astronomical_Object*, *http://dbpedia.org/resource/Object*,

http://dbpedia.org/ointology/Planet, and *http://dbpedia.org/resource/Minor_planet.* We then make the URI substitutions in the extracted axioms. For now, we resort to an elementary disambiguation method, by simply relying on a string matching technique, which will be improved in future work, as discussed in Section 6.7. First, we try to query the DBpedia ontology page (*http://dbpedia.org/ontology/[Input Type]*). In case it is not found, we query the DBpedia resources (*http://dbpedia.org/resource/[Input Type]*). This is a very elementary method that we would like to improve based on our previous work in the OKE challenge [66].

With this straightforward method, we can disambiguate many entities with a risk of mapping to a wrong DBpedia URI. This error would be common for instances with the same name; for example, the name John Smith being a common name, there are multiple DBpedia pages and URIs referring to different people with this same name, such as *http://dbpedia.org/resource/John_Smith_(footballer,_born_1921)*, *http://dbpedia.org/resource/John_Smith_(explorer)*, etc. We can see that our method will not find these links, and would simply refer to *http://dbpedia.org/resource/John_Smith* in all cases, which represent *John Smith*'s disambiguation page.

6.5 Evaluation and Results

We evaluate separately the filtering of instances and classes (6.5.1), the axiom learning process for classes (6.5.2), the type extraction for instances (6.5.3), and the disambiguation (6.5.4). Each one of these modules was evaluated manually. The manual evaluations involved six evaluators that we divided in three groups of two, so each manual evaluation totaled three datasets assessed by two evaluators. These various datasets can be downloaded on our Web site[18]. In Section 6.5.5 we discuss the evaluators' agreement for each dataset.

Also, in some sections, we present additional evaluations; in Section 6.5.1, we compare our result to the output of LExO [87]. In Section 6.5.3, we present the results of the OKE challenge evaluation in which we participated and which evaluates the task of type extraction. In Section 6.5.3, we also conduct an additional automatic evaluation based on DBpedia.

---

[18] www.westlab.polymtl.ca/AXIOpediaWebApp/Evaluation_Results.zip

6.5.1 Evaluation of the process of distinguishing Instances and Classes

Each dataset consists of a random set of 50 resources. To obtain them we randomly selected 25 resources from Wikipedia that we annotated as instances and 25 resources that we annotated as classes.

We asked our evaluators to determine if the resources represent instances or classes, or to indicate that they are uncertain if applicable. By comparing the evaluators results, we were able to build a gold standard; when both annotators agreed on the type of the instance, the type was stored in the gold standard. In case of a disagreement, the entity was annotated by one of the authors of this paper who is a Semantic Web expert. The selected type was the one chosen by a majority of annotators. The results are shown in Table 6.3.

Table 6.3 Results for the Wikipedia Pages Filtering

	Accuracy
	Dataset 1
Classes	96.00%
Instances	84.00%
Average	90.00%
	Dataset 2
Classes	96.00%
Instances	84.00%
Average	90.00%
	Dataset 3
Classes	84.00%
Instances	88.00%
Average	86.00%
	Average
Classes	92.00%
Instances	85.33%
Average	88.67%

As Table 6.3 shows, on average we filtered correctly 88.67% of the resources. We notice that usually the accuracy for the classes are higher than the accuracy for the instances (respectively 92.00% versus 85.33%), which indicates that classes are usually correctly spotted, and there is a larger number of pages representing classes that are annotated as instances instead of classes.

6.5.2 Evaluation of the Axiom Extraction

a) Manual Evaluation of the Axiom Extraction

For this evaluation, we created the datasets based on the results of the page filtering evaluation (6.5.1). For each dataset, we isolated the pages annotated as classes by both evaluators and created new datasets for this evaluation; thus, the dataset 1 consists of 17 resources, dataset 2 consists of 15 resources, and dataset 3 consists of 19 resources. In each evaluation, we presented the Wikipedia resources, their definitions and the generated axioms to the evaluators.

Since there is no standard way to evaluate an axiom in the state of the art, we established a score to express the quality of the axiom. For each class, based on the class' name, definition and built axioms, the evaluator indicated for every axiom in their dataset, the values of the numbers of correct, incorrect and missing classes, predicates and logical operators (union, intersection, etc.). Afterwards, for each evaluator we assessed the correctness of the axioms using standard precision and recall taking into account named classes, predicates, and operators. The final axioms' score metrics are calculated for each evaluator by computing the average of the classes, instances and operators' metrics. The precision and recall on the datasets are then computed as the average precisions and recalls of both evaluators. The results are shown in Table 6.4.

Table 6.4 Result for Manual Evaluation of the Built Axioms

	Precision	**Recall**
	Dataset 1	
Classes	69.10%	75.95%
Predicates	51.57%	33.10%
Operators	93.63%	73.75%
Final Axioms' Score	**71.43%**	**60.93%**
	Dataset 2	
Classes	83.89%	93.33%
Predicates	76.11%	74.39%
Operators	91.94%	91.33%
Final Axioms' Score	**83.98%**	**86.35%**
	Dataset 3	
Classes	65.29%	74.19%
Predicates	65.35%	61.54%
Final Axioms' Score	88.46%	88.46%
Final Axioms' Score	**73.04%**	**74.73%**
	Average	
Classes	72.76%	81.16%
Predicates	64.34%	56.34%
Operators	91.34%	84.51%
Final Axioms' Score	**76.15%**	**74.00%**

As shown in Table 6.4, we were able to get an average of 76.15% for the precision and 74.00% for the recall.

We examined and analyzed the evaluations' results more thoroughly to determine the strengths and weaknesses of our generated axioms. We noticed that for short simple sentences with no ambiguities, we generate perfect axioms with very high precision and recall values that are up to 100%. This is the case for sentences like *A semmelwrap is a Swedish pastry*, or even more complex sentences like *Bogobe jwa lerotse is a porridge that has a wonderfully subtle flavour given to it by*

the lerotse melon. For longer sentences, like *Moorland or moor is a type of habitat found in upland areas in temperate grasslands, savannas, and shrublands and montane grasslands and shrublands biomes, characterised by low-growing vegetation on acidic soils,* we usually have a lower recall when we do not extract all the information of the sentence. However, the value of the precision does not seem to be affected by the sentences' length, and can remain high if the sentence has no grammatical ambiguities. But longer sentences usually have more complex grammatical structures that can thus contain grammatical ambiguities like *An amusement park is a collection of rides and other entertainment attractions assembled for the purpose of entertaining a fairly large group of people*. Building an axiom for this type of sentences based on their grammatical structure is trickier as they often contain some errors in the parsing process. We can note errors like *purpose of entertaining a fairly large group of people* where entertaining is considered as an adjective, which leads to the creation of the class E*ntertainingFairlyLargeGroup* instead of *FairlyLargeGroup*.

Another error comes from the presence of pronouns in the sentences, like *In telecommunications, a repeater is an electronic device that receives a signal and retransmits it*, in which we do not consider the pronouns.

b) Axioms' Comparison with LExO

The authors of LExO conducted an evaluation on 13 axioms generated automatically by their system [87]. The sentences used are selected from real sources; The first 8 sentences are extracted from a fishery glossary provided by the Food and Agriculture Organization (FAO) of the United Nations within the NeOn project[19]. The next 5 sentences are the corresponding Wikipedia definitions of classes chosen from the Proton ontology [33] (developed in the SEKT project[20]).

We executed our approach on this dataset, in order to compare the generated axioms with LExO's axioms. The 13 sentences are presented in annex II, each followed by LExO's output and our output in this order. Hereafter, we present the six axioms used in our discussion:

---

[19] http://www.neon-project.org

[20] http://sekt-project.com

1- Data are facts that result from measurements or observations.

Data ≡ (Fact ∩ ∃result_from.(Measurement ∪ Observation))

Data ⊆ Facts ∩ ∃result.(∃from.(Measurement ∪ Observation))

2- An internal rate of return is a financial or economic indicator of the net benefits expected from a project or enterprise, expressed as a percentage.

InternalRateOfReturn ≡ ((Financial ∪ Economic) ∩ indicator ∩ ∃of.(Net ∩ Benefit ∩ ∃expected_from.(Project ∪ Enterprise)) ∩ ∃expressed_as.Percentage)

InternatRateOfReturn ⊆ ∃FinancialOrEconomicIndicatorOf.NetBenefits ∩ ∃expected.(∃from.(Project ∪ Entreprise))

4- A juvenile is a young fish or an animal that has not reached sexual maturity.

Juvenile ≡ (Young ∩ (Fish ∪ Animal) ∩ ¬∃reached.(Sexual ∩ Maturity))

Juvenile ⊆ (YoungFish ∪ Animal) ∩ ¬∃reached.SexualMaturity

6- A pair trawling is a bottom or mid-water trawling by two vessels towing the same net.

PairTrawling ≡ ((Bottom ∪ MidWater) ∩ Trawling ∩ =2by.(Vessel ∩ ∃tow.(Same ∩ Net)))

PairTrawling ⊆ Trawling ∩ (Bottom ∪ Mid-Water) ∩ ∃towing.(∃by.Vessels)

9- Vehicles are non-living means of transportation.

Vehicle ≡ (¬Living ∩ Means ∩ ∃of.Transportation)

Vehicles ⊆ ∃non-LivingMeansOf.Transportation

12- An island or isle is any piece of land that is completely surrounded by water.

(Island ∪ Isle) ≡ (Piece ∩ ∃of.Land ∩ ∃completely_surrounded_by.Water)

Island ≡ Isle ⊆ ∃pieceOf.Land ∩ ∃completelySurrounded.(∃by.Water)

By analyzing our results (annex II), we can see that, as mentioned earlier, shorter sentences are more likely to be better defined than longer sentences (example 2) or sentences with an ambiguous grammatical structure (example 6).

By comparing LExO's output with ours, we can clearly notice that we process differently the creation of the axioms. One of the most obvious differences is the way we process modifiers and

adjectives. As explained in [87], we can distinguish between three types of adjectives. Subsective adjectives express a subclass relations like $YoungFish \subseteq Fish$, intersective adjectives express an intersection like $SexualMaturity \subseteq (Sexual \cap Maturity)$, and privative adjectives express disjunction such as $FakeFish \subseteq \neg Fish$.

However, automatically distinguishing between these three types is a challenging task; we thus choose one unique way of representing all adjectives for now, but we would like to take into account the 3 types of adjectives in our future work. LExO interprets all adjectives as being intersective, whereas we assume adjectives to be subsective. In that matter, LExO creates an intersection between the adjectives in the axiom, whereas we detect the groups to aggregate and create a related taxonomy. For example, in 4, we create the classes *YoungFish* and *SexualMaturity* which are respectively subclasses of *Fish* and *Maturity* instead of using $(Young \cap Fish)$ and $(Sexual \cap Maturity)$.

Another difference is the prepositions following verbs. LExO aggregates the prepositions to the verbs preceding them, whereas we create a separate predicate with the preposition. For example, in 1, LExO creates the predicate $result_from.(Measurement \cup Observation)$, while we create $result.(\exists from.(Measurement \cup Observation))$. As for the preposition *of* in nominal groups, LExO processes these cases by creating separate predicates, while we aggregate it with its nominal group and before creating a predicate. Our approach is favorable in cases like (9), where *means of transportation* is represented as $meansOf.Transportation$ by our approach against LExO's representation $(Means \cap of.Transportation)$.

We also distinguish between two uses of the conjunction *or*; In some cases, (10 and 12), the conjunction *or* does not express a union in the sentence, but rather an equivalence relation. For example, in (12), we extract the equivalence $(Island \equiv Isle)$ from *island or isle* instead of $(Island \cup Isle)$ like LExO does.

Finally, we do not take into account cardinality constraint (4,6) for the moment, which LExO does. However, it is also important to note that our approach generates axioms in which classes (and instances) are mapped to DBpedia when possible, which is not something LExO does, and that our approach is applied to all Wikipedia, with the purpose of enriching DBpedia.

6.5.3 Type Extraction Evaluation

a) Statistics based on DBpedia

We generated statistic for type extraction from DBpedia. We randomly chose a set of 1000 instances from Wikipedia, for which we extracted the types from Wikipedia and compared them with the types already in DBpedia. The objective of these statistics was to compare the extracted types with the DBpedia types, and to assess our contribution to the enrichment of DBpedia, by examining how many extracted types were already in DBpedia, how many were missing from DBpedia, and how many un-typed pages we were able to type. The results of the evaluation are presented in Table 6.5.

Table 6.5 Result for the Types' Extraction for the Evaluation based on DBpedia

	Results
Number of processed instances	1000
Number of instances that did not have a DBpedia resource	62
Number of instances that did not have a type in DBpedia	401
Number of types extracted	530
Number of untyped instances that we typed	152
Number of types extracted that already existed in DBpedia	100

We notice that among the 1000 instances, 62 did not have a page in DBpedia, 401 instances did not have any type in DBpedia, meaning that 599 were already linked to at least one type with a *rdf:type* relation (note that some of these may be typed by vague or general types like *dbo:Location* or *dbo:Person*). Among the untyped instances, we were able to extract a type for 152 among 401, which represents 37.91% of the untyped instances.

Also, 100 of the 530 extracted *rdf:type* statements already existed in DBpedia which represents 18.87%. Note that we only compare our types with the DBpedia types without considering the aligned types from other knowledge bases.

These statistics allow us to have an overall idea of our contribution to the type extraction task for DBpedia. However, in order to judge the quality of the extracted information in this task, we refer

to the evaluation made by the OKE challenge in addition to a manual evaluation we conducted, which will be presented in the following sections.

b) OKE Evaluation

As mentioned above, a part of this work is an extension of a previous work in the context of the second task of the 2016 OKE challenge. This task is divided in two subtasks. We only consider the evaluation of the first subtask, that evaluates the creation of a *rdf:type* statements for entities based on their textual definition from Wikipedia. The train[21] and test[22] datasets are provided in NIF[23] format, and are composed of entities and their respective definitions extracted from Wikipedia, along with the expected extracted types and the expected aligned types from DOLCE+DnS Ultra Lite; the train dataset consists of 200 entities, and the test dataset consists of 50 entities.

The results for the type extraction subtask for the 2016 train and test datasets are presented in Table 6.6.

Table 6.6 Result for the OKE 2016 Evaluation

	Micro			**Macro**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
2016 Train Dataset	82.32%	75.81%	78.93%	82.32%	78.96%	80.05%
2016 Test Dataset	81.63%	73.39%	77.29%	80.81%	76.60%	77.95%

As shown in Table 6.7, we obtain high values of precision and recall (>75%) for both datasets. Our system also obtained the best result among the participants of the challenge in 2016. These results

[21] https://github.com/anuzzolese/oke-challenge-2016/blob/master/GoldStandard_sampleData/task2/dataset_task_2.ttl

[22] https://github.com/anuzzolese/oke-challenge-2016/blob/master/evaluation-data/task2/evaluation-dataset-task2.ttl

[23] http://persistence.uni-leipzig.org/nlp2rdf/

are presented and further discussed in [66]. Note that in the OKE challenge we did not execute the Hearst patterns, nor the aggregation of the types with their modifiers and with the preposition “of”.

c) Manual Evaluation of the Type Extraction

For this evaluation, we created the datasets based on the results of the page filtering evaluation (6.5.1). For each dataset, we extracted the resources annotated as instances by both evaluators and created new datasets for this evaluation; thus, dataset 1 consists of 21 resources, dataset 2 consists of 15 resources, and dataset 3 consists of 16 resources. In each evaluation, we presented the Wikipedia resources, their definitions and the extracted types to the evaluators.

For each instance in their dataset, we asked the evaluators to assess the number of correct types extracted and the number of missing types that were not extracted from the sentence. We computed the average of the precision and recall of both evaluators for every dataset. The results are presented in Table 6.7.

Table 6.7 Result for the Types’ Extraction Manual Evaluation

	Precision	**Recall**
Dataset 1	84.09%	67.26%
Dataset 2	83.33%	73.52%
Dataset 3	90.63%	85.29%
Average of the Datasets 1, 2 and 3	**86.02%**	**75.37%**

We obtain an average precision of 86.02% and average recall of 75.37%. Again, most notable errors come from grammatical ambiguities.

6.5.4 Disambiguation Evaluation

In the disambiguation process, we link the extracted entities with existing DBpedia URIs when possible. In this section, we evaluate this process, and create three datasets in which we present between 115 and 125 entities with their respective disambiguated DBpedia URIs. We compute the average of the precision and recall of both evaluators for each dataset. The results are presented in Table 6.8 for the three datasets of this evaluation.

Table 6.8 Result for the Disambiguation Manual Evaluation

	Precision	**Recall**
Dataset 1	69.57%	42.33%
Dataset 2	81.56%	53.21%
Dataset 3	67.20%	30.30%
Average of the Datasets 1, 2 and 3	**72.78%**	**41.95%**

For this section, we obtain an average precision of 72.78% and an average recall of 41.95%. As explained earlier, we only rely on an exact string matching technique, which is why most errors occur when there is more than one entity with the same string. In those cases, there is usually a DBpedia URI representing a disambiguation page, which we use instead of using the URI representing the correct class.

6.5.5 Inter-Annotator Agreement

As explained above, due to the restricted number of evaluators, our manual evaluation only totaled two annotators per dataset for three different datasets. To measure the reliability of the evaluations, we resort to a comparison between the results of the evaluations conducted on the same datasets, and thus, we judge of the inter-annotator agreement. For every manual evaluation, we measure the agreement between the two annotators by measuring the observed agreement and the agreement expected by chance, and calculate the kappa coefficient [34] on which we base our judgment. For the evaluation of the agreement on the generated axiom, we calculate the mathematic difference

between the different items (classes, predicates and operators) and present the average of the differences. The results are presented in tables 6.9, 6.10, 6.11, 6.12.

Table 6.9 Agreement for the Page Filtering Evaluation

	Observed Agreement	**Agreement Expected by Chance**	**Kappa Coefficient**	**Strength of Agreement**
Dataset 1	90.00%	46.40%	0.813	Good
Dataset 2	84.00%	49.40%	0.684	Good
Dataset 3	84.00%	47.52%	0.695	Good

For the filtering evaluation (table 6.9), the kappa coefficient varies between 0.695 and 0.813 on the three datasets. These values represent a good agreement between each pair of annotators in the three datasets.

Table 6.10 Agreement for the Axioms' Evaluation

	Difference on the Number of Correct Classes	Difference on the Number of Incorrect Classes	Difference on the Number of Missing Classes	Difference on the Number of Correct Predicates	Difference on the Number of Incorrect Predicates	Difference on the Number of Missing Predicates	Difference on the Number of Correct Operators	Difference on the Number of Incorrect Operators	Difference on the Number of Missing Operators
Dataset 1	5.588	1.706	1.765	0.647	0.529	1.353	0.588	0.235	1.235
Dataset 2	0.933	0.400	0.133	1.067	0.533	0.733	0.933	0.467	0.733
Dataset 3	6.211	0.684	0.737	1.947	0.895	0.947	0.737	0.474	0.158

For the axioms' generation (table 6.10), it was very hard to establish an annotator agreement for each single part of the axiom and for the complete axiom. Thus, we simply compared the different ratings of the evaluators by calculating the average of their differences. The obtained value is the

difference between the number of items accepted by one evaluator versus the other. For example, on the first dataset the difference on the number of correct classes is 5.58, which means that one of the evaluators on average evaluated 5 additional classes as correct compared to the second evaluator. Two values stand out in table 6.10 for the number of correct classes in dataset 1 and 3 (5.588 and 6.211). The reason why the values are so high for this category is that in both these datasets, one of the evaluators counted not only the classes in the axiom generated, but also the classes in the extracted taxonomical relations. These numbers can also be explained by the fact that axioms are harder to evaluate since there is no strict rule describing the correct way of expressing an axiom based on a sentence, so axioms can be defined differently depending on how the evaluator interprets the sentence.

Table 6.11 Agreement for the Type Extraction Manual Evaluation

	Observed Agreement	**Agreement Expected by Chance**	**Kappa Coefficient**	**Strength of Agreement**
Dataset 1	95.65%	63.52%	0.881	Very Good
Dataset 2	92.86%	70.41%	0.759	Good
Dataset 3	81.25%	81.25%	0.000	Poor

For the type extraction (table 6.11), the kappa value is 0.881 on the first dataset, indicating a very high agreement between the annotators, and 0.759 on the second dataset, indicating a good agreement. However, on the third dataset, the kappa value is 0 meaning that the agreement is very poor between the two annotators on this dataset, even though the observed agreement is high and has a value of 81.25%. The kappa coefficient takes the value of 0 since one of the evaluators for this dataset agreed with all the extracted types; in that case, the agreement expected by chance takes the value of the observed agreement, and the kappa coefficient is null. This is one of the limitations of the kappa coefficient.

Table 6.12 Agreement for the Disambiguation Manual Evaluation

	Observed Agreement	**Agreement Expected by Chance**	**Kappa Coefficient**	**Strength of Agreement**
Dataset 1	66.67%	56.89%	0.227	Fair

Dataset 2	72.95%	66.69%	0.188	Poor
Dataset 3	81.60%	54.20%	0.598	Moderate

Finally, for the disambiguation evaluation (table 6.12), the kappa values are respectively 0.227, 0.188, and 0.072 on the datasets 1, 2 and 3, indicating that the agreement is fair for the first dataset, poor for the second, and moderate for the last one. The disagreement comes from the fact that some evaluators annotated the disambiguation as correct even when the URL represented a disambiguation page, while some evaluators considered that as incorrect.

6.6 Conclusion and Future Work

In conclusion, we presented an approach to extract axioms and types from natural text definitions in Wikipedia. We give access to our Web service to generate axioms from textual definitions. We also generated a complete ontology, which can be accessed on our website and is based on the full English version of Wikipedia (November 2016), and integrates to DBpedia by using DBpedia URIs when possible.

There are however some limitations to our work. First, the filtering process to distinguish classes and instances attain acceptable results (with a precision and recall of 80% as shown in section 6.5.1). Yet, these results could be improved by analyzing other information such as the DBpedia triples of the resources or the resources' Wikipedia categories. For example, if a resource contains a birth date we would infer that it is an instance. We could also resort to other datasets like OpenCyc [94] or WordNet [95] where the resources are already annotated as classes or instances.

Also, using statistical learning to extract automatically the recurrent patterns would contribute to the improvement of our results. It will enable us to identify patterns which are less frequent or which are more difficult to detect manually. In our future work, we plan to use anaphora resolution, to disambiguate pronouns with their antecedents before parsing the sentences. We also plan to distinguish between the three types of adjectives (subjective, intersective and privative adjectives).

Finally, we will rely on more sophisticated methods for axiom and predicate disambiguation rather than the simple string matching proposed here. This will enable a better integration of AXIOpedia with DBpedia.

Acknowledgement. This research has been funded by the NSERC Discovery Grant Program.

6.7 References

L. Bühmann and J. Lehmann, "Pattern Based Knowledge Base Enrichment," in International Semantic Web Conference. Springer, Berlin Heidelberg, 2013.

L. Font, A. Zouaq and M. Gagnon, "Assessing the quality of domain concepts descriptions in DBpedia," in Signal-Image Technology & Internet-Based Systems (SITIS). 11th International Conference on. IEEE., 2015.

W. Wong, W. Liu and M. Bennamoun, "Ontology learning from text: A look back and into the future," ACM Computing Surveys (CSUR), vol. 44, no. 4, p. 20, 2012.

L. Haidar-Ahmad, A. Zouaq and M. Gagnon, "Automatic Extraction of Axioms from Wikipedia Using SPARQL," Springer Link: Lecture Notes in Computer Science, vol. 9989, pp. 60-64, 2016.

L. Haidar-Ahmad, L. Font, A. Zouaq and M. Gagnon, "Entity typing and linking using sparql patterns and DBpedia.," Semantic Web Evaluation Challenge. Springer International Publishing., vol. 641, pp. 61-75, 2016.

M. Sabou, E. Blomqvist, T. Di Noia, H. Sack and T. Pellegrini, "User Driven Quality Evaluation of DBpedia," in Proceedings of the 9th International Conference on Semantic Systems, Austria, 2013.

C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak and S. Hellmann, "DBpedia Crystalization Point for the Web of Data," Web Semantics: science, services and agents on the world wide web 7.3, vol. 7, no. 3, pp. 154-165, 2009.

F. Orlandi and A. Passant, "Modelling provenance of DBpedia resources using Wikipedia contributions," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 9, no. 2, pp. 149-164, 2011.

G. Töpper, M. Knuth and H. Sack, "DBpedia ontology enrichment for inconsistency detection," in Proceedings of the 8th International Conference on Semantic Systems. ACM., 2012.

M. Morsey, J. Lehmann, S. Auer, C. Stadler and S. Hellmann, "Dbpedia and the live extraction of structured data from wikipedia," Program, vol. 46, no. 2, pp. 157-181, 2012.

J. Waitelonis, N. Ludwig, M. Knuth and H. Sack, "Whoknows? evaluating linked data heuristics with a quiz that cleans up dbpedia," nteractive Technology and Smart Education, vol. 8, no. 4, pp. 236-248, 2011.

K. Nakayama, M. Pei, M. Erdmann, M. Ito, M. Shirakawa, T. Hara and S. Nishio, "Wikipedia Mining Wikipedia as a Corpus for Knowledge Extraction," 2008.

O. Medelyan, D. Milne, C. Legg and I. H. Witten, "Mining meaning from Wikipedia," International Journal of Human-Computer Studies, vol. 67, no. 9, pp. 716-754, 2009.

P. Buitelaar, P. Cimiano and B. Magnini, "Ontology learning from text: An overview," in Ontology learning from text: Methods, evaluation and applications, IOS Press, 2005, pp. 3-12.

S. P. Ponzetto and M. Strube, "Wikitaxonomy : A Large Scale Knowledge Resource," ECAI, vol. 178, pp. 751-752, 2008.

M. Röder, R. Usbeck, R. Speck and A.-C. Ngonga Ngomo, "CETUS – A Baseline Approach to Type Extraction," Springer: Communications in Computer and Information Science, vol. 548, pp. 16-27, 2016.

J. Gao and S. Mazumda, "Exploiting Linked Open Data to Uncover Entity Types," Springer : Communications in Computer and Information Science, vol. 548, pp. 51-62, 2016.

S. Consoli and D. Reforgiato Recupero, "Using FRED for Named Entity Resolution, Linking and Typing for Knowledge Base Population," Springer : Communications in Computer and Information Science, vol. 548, pp. 40-50, 2016.

S. Faralli and S. P. Ponzetto, "DWS at the 2016 Open Knowledge Extraction Challenge: A Hearst-Like Pattern-Based Approach to Hypernym Extraction and Class Induction," Springer : Communications in Computer and Information Science, vol. 641, pp. 48-60, 2016.

J. Bos, "Wide-coverage semantic analysis with boxer," Proceedings of the 2008 Conference on Semantics in Text Processing - Association for Computational Linguistics, pp. 277-286, 2008.

M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," Proceedings of the 14th conference on Computational linguistics - Association for Computational Linguistic, vol. 2, pp. 539-545, 1992.

T. Kliegr and O. Zamazal, "LHD 2.0: A text mining approach to typing entities in knowledge graphs," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 39, pp. 47-61, 2016.

Q. Liu, K. Xu, L. Zhang, H. Wang, Y. Yu and Y. Pan, "Catriple: Extracting triples from wikipedia categories," in Asian Semantic Web Conference. Springer., Berlin Heidelberg, 2008.

F. Wu and D. S. Weld, "Open information extraction using Wikipedia," in Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics., 2010.

A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead and S. Soderland, "Textrunner: open information extraction on the web," in Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, . Association for Computational Linguistics., 2007.

G. Wang, Y. Yu and H. Zhu, "Pore: Positive-only relation extraction from wikipedia text," in The Semantic Web. Springer., Berlin Heidelberg, 2007.

J. Völker, P. Hitzler and P. Cimiano, "Acquisition of OWL DL Axioms from Lexical Resources," Springer : Lecture Notes in Computer Science, vol. 4519, pp. 670-685, 2007.

J. Völker and M. Niepert, "Statistical Schema Induction," Extended Semantic Web Conference - Springer Berlin Heidelberg, 2011.

L. Bühmann and J. Lehmann, "Pattern Based Knowledge Base Enrichment," International Semantic Web Conference - Springer Berlin Heidelberg, pp. 33-48, 2013.

Z. Cäcilia, V. Nastase and M. Strube, "Distinguishing between Instances and Classes in the Wikipedia Taxonomy," European Semantic Web Conference - Springer Berlin Heidelberg, pp. 376-387, 2008.

J. Rose Finkel, T. Grenager and C. Manning, "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling," in Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), 2005.

M.-C. De Marneffe and C. D. Manning, "The Stanford typed dependencies representation," in Proc. of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation. ACL., 2008.

I. Terziev, A. Kiryakov and D. Manov., Base Upper-Level Ontology (BULO) guidance. SEKT., Ontotext Lab, Sirma AI EAD (Ltd.), 2004.

J. Cohen, "A coefficient of agreement for nominal scales," Educational and psychological measurement, vol. 20, no. 1, pp. 37-46, 1960.

Cycorp., "OpenCyc," [Online]. Available: http://opencyc.org/. [Accessed 15 February 2017].

G. A. Miller, "WordNet: a lexical database for English," Communications of the ACM, vol. 38, no. 11, pp. 39-41, 1995.

D. Yarowsky, "Word-sense disambiguation using statistical models of Roget's categories trained on large corpora," in Proceedings of the 14th conference on Computational linguistics-Volume 2. Association for Computational Linguistics., 1992.

S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z. Ives, "Dbpedia: A nucleus for a web of open data," in In The semantic web, Berlin Heidelber, Springer, 2007, pp. 722-735.

F. M. Suchanek, G. Kasneci and G. Weikum, "Yago: A large ontology from wikipedia and wordnet," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 6, no. 3, pp. 203-217, 2008.

G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information processing & management, vol. 24, no. 5, pp. 513-523, 1988.

K. T. Frantzi and S. Ananiadou, "The C-value/NC-value domain-independent method for multi-word term extraction," Journal of Natural Language Processing, vol. 6, no. 3, pp. 145-179, 1999.

P. Pantel and D. Lin, "A statistical corpus-based term extractor," in Conference of the Canadian Society for Computational Studies of Intelligence. Springer., Berlin Heidelberg, 2001.

V. I. Spitkovsky and A. X. Chang, "A Cross-Lingual Dictionary for English Wikipedia Concepts," LREC, pp. 3168-3175, 2012.

D. Nguyen, A. Overwijk, C. Hauff, D. R. Trieschnigg, D. Hiemstra and F. De Jong, "WikiTranslate: query translation for cross-lingual information retrieval using only Wikipedia," in Workshop of the Cross-Language Evaluation Forum. Springer., Berlin Heidelberg, 2008.

D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in Proceedings of the 33rd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics., 1995.

O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld and A. Yates, "Web-scale information extraction in knowitall:(preliminary results)," in Proceedings of the 13th international conference on World Wide Web. ACM., 2004.

R. Evans and S. Street, "A framework for named entity recognition in the open domain," Recent Advances in Natural Language Processing III: Selected Papers from RANLP, vol. 260, no. 110, pp. 267-274, 2003.

6.8 Annex

Annex I

Table 6.13 Patterns and their mapped axioms

Patterns	**Corresponding Axioms**
(1) :neg A B *In Foreign exchange market, synthetic currency pair or synthetic cross currency pair is an artificial currency pair which generally is* ***not available*** *in market but one needs to trade across those pairs.*	$A \equiv \neg B$ *NotAvailable* $\equiv \neg$*Available*
(2) :cc :cc:or OR A B X :nsubj *A* ***natural satellite or moon*** *is an astronomical object.*	A ≡ B *NaturalSatellite* ≡*Moon*
(3) :nmod:of A B *Vehicles are non-living* ***means of transportation****.*	AOfB ≡ ∃AOf.B *NonLivingMeansOfTransportation* ≡ *∃NonLivingMeansOf.Transportation*
(4) :nsubj :cop	A ⊆ B

A B C ***A computer is a device*** *that can be* instructed *to carry out an arbitrary set of arithmetic or* logical *operations automatically.*	*Computer ⊆ Device*
(5) :nsubj :cop :dobj A B C D E :acl *A number is an abstract entity* ***representing a count or measurement****.*	A ⊆ ∃D.E *Number ⊆ ∃representing.CountOrMeasurement*
(6) :nsubj :cop :nmod:PREP A B C D E :acl *A lake is a body of water* ***surrounded by land****.*	A ⊆ ∃D.(∃PREP.E) *Lake ⊆ ∃surrounded.(∃by.Water)*
(7) :nsubj :cop :xcomp :dobj A B C D E F TO :acl :mark *A project is a temporary endeavor* ***undertaken to create a unique product or service****.*	A ⊆ ∃D.(∃TO_E.F) *Project ⊆ ∃undertaken.(∃toCreate.UniqueProductOrService)*
(8) :nsubj :cop :xcomp :dobj A B C D E F TO G :acl :mark :nmod:PREP *A project is a temporary endeavor* ***undertaken to create a unique product or service with a purpose****.*	A ⊆ ∃D.(∃TO_E.(∃PREP.G) *Project ⊆ ∃undertaken.(∃toCreate.(∃with.Purpose)*
(9) :nsubj :cop :xcomp A B C D E TO :acl :mark :dobj MINUS : E G *A sentence is a decree* ***used to punish****.*	A ⊆ ∃D.(∃TO_E.⊤) *Sentence ⊆ ∃used.(∃toPunish. ⊤)*
(10) :nsubj :cop :advmod A B C D E F	A ⊆ ∃E.(∃D.F)

:acl:relcl :nsubj *An airport is a facility* ***where aircraft can take off and land****.*	*Airport ⊆ ∃where.(∃CanTakeOffAndLand.Aircraft)*
(11) :nsubj :cop :dobj A B C D E :acl:relcl :nsubj/:nsubjpass *Theatre or theater is a collaborative form of fine art that* ***uses live performers****, to present the experience of a real or imagined event before an audience in a specific place, often a stage.*	A ⊆ ∃D.E *Theatre ⊆ ∃uses.LivePerformers*
(12) :nsubj :cop :nmod :PREP A B C D E :acl:relcl :nsubj/:nsubjpass *A valley is a landform,* ***which can range from a few square miles to hundreds or even thousands of square miles in area****.*	A ⊆ ∃D.(∃PREP.E) *Valley ⊆ ∃canRange.(∃from.FewSquareMiles)* *Valley ⊆ ∃canRange.(∃to.HundredsOrEvenThousandsOfSquareMiles)*
(13) :nsubj :cop :dobj :nmod:PREP A B C D E F :acl:relcl :nsubj/:nsubjpass *Theatre or theater is a collaborative form of fine art that* ***uses live performers before an audience*** *in a specific place, often a stage.*	A ⊆ D_PREP.F D_PREP ⊆ D *Theatre ⊆ ∃uses.(∃before.Audience)*
(14) :nsubj :cop :advcl/:xcomp :mark A B C D E TO F :acl:relcl :dobj :nsubj/:nsubjpass *Theatre or theater is a collaborative form of fine art that* ***uses*** *live performers,* ***to present the experience of a real or imagined event*** *before an audience in a specific place, often a stage.*	A ⊆ ∃D.(∃TO_E.F) *Theatre ⊆ ∃uses.(∃toPresent.ExperienceOfRealOrImaginesEvent)*
(15) :nsubj :cop :advcl/:xcomp :mark A B C D E TO F	A ⊆ ∃D.(∃TO_E.(∃PREP.F)

:acl:relcl :nmod:PREP :nsubj/:nsubjpass *Theatre or theater is a collaborative form of fine art that **uses** live performers, **to present** the experience of a real or imagined event **before an audience** in a specific place, often a stage.*	*Theatre ⊆ ∃uses.(∃toPresent.(∃before.Audiance))*
(16) :nsubj :cop :advcl/:xcomp :mark A B C D E TO :acl:relcl :dobj MINUS : E F *Theatre or theater is a collaborative form of fine art that **enables** live performers **to perform**.*	A ⊆ ∃D.(∃TO_E.⊤) *Theatre ⊆ ∃enables.(∃toPerform.⊤)*
(17) :nsubj :cop A B C D :nmod:in ***In*** geography*, a plain is a flat area.*	A ⊆ ∃in.D *Plain ⊆ ∃in.Geography*
(18) :acl :nmod:PREP A B C *Eucalyptus crucis, comonly **known as the silver mallee**, is a eucalypt that is native to Western Australia.*	A ⊆ ∃B.∃PREP.C *EucalyptusCrucis ⊆ ∃known.(∃as.SilverMallee)*
(19) :nsubjpass :auxpass :mark A B C D TO :xcomp :dobj MINUS : D E *A deadly weapon **is used to kill** in a murder.*	A ⊆ ∃B.(∃TO_D.⊤) *DeadlyWeapon ⊆ ∃used.(∃toKill.⊤)*
(20) :nsubjpass :auxpass :mark A B C D TO E :xcomp :dobj *The term sitelet is **used to describe various web pages** or web page elements.*	A ⊆ ∃B.(∃TO_D.E)

	TermSitelet ⊆ ∃used.(∃toDescribe.VariousWebPages)
(21) :nsubjpass :auxpass A B C D :nmod:PREP *Ghosts are believed to be wandering souls and **are thought by Vietnamese people** to affect their daily lives.*	A ⊆ ∃B.(∃PREP.D) *Ghosts ⊆ ∃thought.(∃by.VietnamesePeople)*
(22) :nsubjpass :auxpass :mark A B C D TO E :xcomp :nmod:PREP *Ghosts are sometimes **believed to haunt people for revenge**.*	A ⊆ ∃B.(∃TO_D.(∃PREP.E)) *Ghosts ⊆ ∃believed.(∃toHaunt.(∃for.Revenge))*
(23) :nsubj :cop A B C D :nmod:PREP ***A cauldron is a large metal pot for cooking and/or boiling** over an open fire.*	A ⊆ ∃PREP.D *Cauldron ⊆ ∃for.CookingOrBoiling*
(24) :nsubj :dobj A B C :cop MINUS : B D :mark MINUS : B TO E :xcomp *In the pantheon of Mongolian shamanism, **tngri constitute the highest class of divinities** and are attested in sources going back to the 13th century.*	A ⊆ ∃B.C *Tngri ⊆ ∃constitute.HighestClassOfDivinities*
(25) :nsubj :nmod:PREP A B C :cop MINUS : B D :mark	A ⊆ ∃B.(∃C.D)

MINUS : B TO E :xcomp *An employee* ***contributes with labor and expertise*** *to an endeavor of an employer.*	*Employee ⊆ ∃contributes.(∃with.LaborAndExpertise)*
(26) :nsubj :dobj :nmod:PREP A B C D :cop MINUS : B D :mark MINUS : B TO E :xcomp ***Tngri constitute the highest class of divinities in the pantheon of Mongolian shamanism.***	A ⊆ ∃B.(∃C.D) *Tngri ⊆ ∃constitute.(∃in.PantheonOfMongolianShamanism)*
(27) :nsubj :nmod:to/of A B C :cop MINUS : B D *The term molecular recognition* ***refers to the specific interaction*** *between two or more molecules through noncovalent bonding.*	A ⊆ ∃B.(∃TO/OF.C) *TermMolecularRecognition ⊆ ∃refers.(∃to.SpecificInteraction)*
(28) :nsubj :nmod:to/of :nmod:PREP A B C D :cop MINUS : B E *The term molecular recognition* ***refers to the specific interaction*** *between two or more molecules* ***through noncovalent bonding****.*	A ⊆ ∃B.(∃TO/OF.(∃PREP.D)) *TermMolecularRecognition ⊆ ∃refers.(∃to.(∃through.NonCovalentBonding))*
(29) :nsubj :cop :nsubj/:nsubjpass A B C D WHICH E :nmod:PREP :acl:relcl	A ⊆ ∃PREP_WHICH.(∃E_D.⊤)

:ref :dobj MINUS : B F *A residential area is a land use* ***in which housing predominates****.*	*ResidentialArea* ⊆ *∃inWich.(∃housingPredominates.⊤)*
(30) :nsubj :cop :nsubj/:nsubjpass A B C D WHICH E F :nmod:PREP :dobj :acl:relcl :ref *Drift migration is the phenomenon* ***in which migrating birds are blown off course*** *by the winds while they are in flight.*	A ⊆ ∃PREP_WHICH.(∃E_D.F) *DriftingMigration* ⊆ *∃inWich.(∃migratingBirdsAreBlownOff.Course)*
(31) :nsubj :cop :nsubj/:nsubjpass A B C D WHICH E F :nmod:PREP :nmod:PREP :acl:relcl :ref *A residential area is a land use* ***in which housing dominates by number****.*	A ⊆ ∃PREP_WHICH.(∃E_D.(∃PREP.F)) *ResidentialArea* ⊆ *∃inWich.(∃housingPredominates.(∃by.number))*
(32) :nsubj :cop :nsubj/:nsubjpass :nmod:PREP A B C D WHICH E F G :nmod:PREP :dobj :acl:relcl :ref *Drift migration is the phenomenon* ***in which migrating birds are blown off*** *course* ***by the winds*** *while they are in flight.*	A ⊆ ∃PREP_WHICH.(∃E_D.(∃PREP.G)) *DriftingMigration* ⊆ *∃inWich.(∃migratingBirdsAreBlownOff.(∃by.Winds))*

Table 6.14 Patterns for detecting Instances' Types

Patterns		
(1)	:nsubj	:cop

Entity X Z Type
:nmod:of

Where X = { “name”, “nickname”, “surname”, “alias”, “one” }

Sant'Elmo is the name of both a hill and a fortress in Naples, located near the Certosa di San Martino.
***Entity** = Sant'Elmo ; **Type** = Hill, Fortress*

(2) :nsubj :cop
Entity Type Z

El Oso, released in 1998, is an album by the New York City band Soul Coughing.
***Entity** = El Oso ; **Type** = Album*

(3) :nsubjpass :auxpass
Entity V Z Type
:nmod:as

Bromius in ancient Greece was used as an epithet of Dionysus/Bacchus.
***Entity** = Bromius ; **Type** = Epithet*

(4) x
Entity Type

Where x = { :compound, :amod, :appos, :nn}

The AES11 standard published by the Audio Engineering Society provides a systematic approach to the synchronization of digital audio signals.
***Entity** = AES11 ; **Type** = Standard*

Table 6.15 Hearst Patterns and their mapped axioms

Hearst Patterns	**Corresponding Axioms**
(1) :nmod:as :case A B SUCH AS :amod *A person is **a being, such as a human**, that has certain capacities or attributes constituting personhood.*	B is a A: If B is a Class : B $\subseteq$ A If B is an instance : B rdf:type A *Human $\subseteq$ Being*
(2) :amod :advmod A B ESPECIALLY OR :dep A B ESPECIALLY	B is a A: If B is a Class : B $\subseteq$ A If B is an instance : B rdf:type A

:advmod *Borscht is a sour soup popular in several* ***Eastern European cuisines, especially Ukrainian cuisine****.*	*UkranianCuisine* $\subseteq$ *EasternEuropeanCuisine*
(3) :including A B *Borscht is a sour soup popular in several* ***Eastern European cuisines, including Ukrainian and Russian cuisine****s.*	B is a A: If B is a Class : B $\subseteq$ A If B is an instance : B rdf:type A *UkranianCuisine* $\subseteq$ *EasternEuropeanCuisine* *RussianCuisine* $\subseteq$ *EasternEuropeanCuisine*
(4) :cc:or/and A B OTHER :amod *An amusement park is a collection of* ***rides and other entertainment attractions*** *assembled for the purpose of entertaining a fairly large group of people.*	B is a A: If B is a Class : B $\subseteq$ A If B is an instance : B rdf:type A *Rides* $\subseteq$ *EntertainmentAttractions*

Annex II

1- Data are facts that result from measurements or observations.

Data ≡ (Fact ∩ ∃result_from.(Measurement ∪ Observation))

Data ⊆ Facts ∩ ∃result.(∃from.(Measurement ∪ Observation))

2- An internal rate of return is a financial or economic indicator of the net benefits expected from a project or enterprise, expressed as a percentage.

InternalRateOfReturn ≡ ((Financial ∪ Economic) ∩ indicator ∩ ∃of.(Net ∩ Benefit ∩ ∃expected_from.(Project ∪ Enterprise)) ∩ ∃expressed_as.Percentage)

InternatRateOfReturn ⊆ ∃FinancialOrEconomicIndicatorOf.NetBenefits ∩ ∃expected.(∃from.(Project ∪ Entreprise))

3- A vector is an organism which carries or transmits a pathogen.

Vector ≡ (Organism ∩ (carry ∪ ∃transmit.Pathogen))

Vector ⊆ Organism ∩ ∃carriesOrTransmits.Pathogen

4- A juvenile is a young fish or an animal that has not reached sexual maturity.

Juvenile ≡ (Young ∩ (Fish ∪ Animal) ∩ ¬∃reached.(Sexual ∩ Maturity))

Juvenile ⊆ (YoungFish ∪ Animal) ∩ ¬∃reached.SexualMaturity

5- A tetraploid is a cell or an organism having four sets of chromosomes.

Tetraploid ≡ ((Cell ∪ Organism) ∪ =4having.(Set ∩ ∃of.Chromosomes))

Tetraploid ⊆ (Cell ∪ Organism) ∩ ∃having.(∃setsOf.Chromosomes)

6- A pair trawling is a bottom or mid-water trawling by two vessels towing the same net.

PairTrawling ≡ ((Bottom ∪ MidWater) ∩ Trawling ∩ =2by.(Vessel ∩ ∃tow.(Same ∩ Net)))

PairTrawling ⊆ Trawling ∩ (Bottom ∪ Mid-Water) ∩ ∃towing.(∃by.Vessels)

7- A sustained use is a continuing use without severe or permanent deterioration in the resources.

SustainedUse ≡ (Continuing ∩ Use ∩ ¬∃with.((Severe ∪ Permanent) ∩ Deterioration ∩ ∃in.Resources))

SustainedUse ⊆ Use ∩ ContinuingUse ∩ ∃without.SevereOrPremanentDeterioration

8- A biosphere is the portion of Earth and its atmosphere that can support life.

Biosphere ≡ (Portion ∩ ∃of.((Earth ∪ (Its ∩ Atmosphere)) ∪ ∃can_support.Life))

Biosphere ⊆ ∃portionAndAtmosphereOf.Earth ∩ ∃support.Life

9- Vehicles are non-living means of transportation.

Vehicle ≡ (¬Living ∩ Means ∩ ∃of.Transportation)

Vehicles ⊆ ∃non-LivingMeansOf.Transportation

10- A minister or a secretary is a politician who holds significant public office in a national or regional government.

(Minister ∪ Secretary) ≡ (Politician ∩ ∃holds.((Office ∩ Significant ∩ Public) ∩ ∃in.(Government ∩ (National ∪ Regional))))

Minister ≡ Secretary ⊆ Politician ∩ ∃holds.(SignificantPublicOffice ∩ ∃in.NationalOrRegionalGovernment)

11- A currency is a unit of exchange, facilitating the transfer of goods and services.

Currency ≡ (Unit ∩ ∃of.Exchange ∩ ∃facilitate.(Transfer ∩ ∃of.(Good ∩ Service)))

Currency ⊆ ∃unitOf.Exchange ∩ ∃facilitating.(∃tranferOf.(Goods ∩ Services))

12- An island or isle is any piece of land that is completely surrounded by water.

(Island ∪ Isle) ≡ (Piece ∩ ∃of.Land ∩ ∃completely_surrounded_by.Water)

Island ≡ Isle ⊆ ∃pieceOf.Land ∩ ∃completelySurrounded.(∃by.Water)

13- Days of the week are: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.

DayOfWeek ≡ {Monday,Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}

DaysOfWeek ⊆ SundayMondayTuesdayWednesdayThursdayFridaySaturday

CHAPITRE 7 DISCUSSION GÉNÉRALE

Dans ce chapitre, nous effectuons une discussion générale de ce travail. Nous analysons tout d'abord des résultats obtenus. Dans la section 7.2, nous revenons sur les objectifs et les questions de recherche que nous nous sommes posées au début de notre recherche. Dans la section 7.3, nous discutons des points à améliorer et des travaux futurs, avant de conclure dans la partie 7.4.

7.1 Analyse et discussion des résultats

L'article 3 contient les résultats détaillés de nos évaluations.

Nous générons des axiomes particulièrement bons pour les phrases courtes, concises, ayant une structure grammaticale claire. Pour les phrases plus longues, des erreurs se présentent lorsque les phrases contiennent des ambiguïtés grammaticales, qui causent des erreurs dans la sortie du Stanford Parser, dans l'étape de l'analyse grammaticale.

Similairement, pour les instances, les erreurs dans l'extraction de types proviennent généralement d'erreurs du Stanford Parser causées par une ambigüité grammaticale ou une structure complexe de la phrase.

7.2 Retour sur les objectifs et questions de recherche

Au début de notre travail, nous nous sommes posé les questions suivantes :

- La structure grammaticale d'une phrase serait-elle suffisante pour extraire des axiomes et liens taxonomiques en y identifiant des patrons syntaxiques récurrents?

Notre recherche montre que la structure grammaticale d'une phrase reflète bien les relations que celle-ci contient. En effet, en nous basant uniquement sur les occurrences de patrons retrouvés dans la syntaxe et les relations grammaticales d'une phrase, nous avons réussi à construire un outil capable d'extraire des axiomes, des instances et des liens taxonomiques de cette phrase.

- Quels sont les patrons syntaxiques les plus récurrents dans les définitions extraites de Wikipédia?

Nous avons établi manuellement une liste de patrons syntaxiques les plus récurrents de la langue anglaise; nous comptons 32 patrons pour extraire des axiomes, 4 patrons pour les extraire des types d'instance, en plus des 4 patrons de Hearst. Ces patrons ont été établis en examinant la syntaxe de définitions de Wikipédia, mais ne sont pas spécifiques aux définitions, c.-à-d. qu'on peut les exécuter sur n'importe quelles phrases en anglais.

- Est-il avantageux de profiter de la simplicité de RDF et SPARQL pour la recherche d'occurrences de patrons syntaxiques dans une phrase?

Concernant l'utilisation de SPARQL et RDF, bien que quelques limitations de SPARQL aient rendu le processus plus complexe, la simplicité de ce langage a été un facteur facilitant la tâche d'implémentation des patrons et aurait une valeur considérable quant à l'ajout potentiel de nouveaux patrons.

En plus de fournir un outil pour extraire des axiomes et des types à partir de définitions textuelles, nous contribuons au Web sémantique en offrant une ontologie générée automatiquement à partir des définitions textuelles de la version anglaise complète de Wikipédia. L'ontologie générée s'intègre à DBpedia en utilisant ses URIs lorsque c'est possible. L'ontologie générée est accessible à partir de notre site Web[24]. Nous donnons en plus accès à un outil permettant aux chercheurs de générer des axiomes pour n'importe quelle définition textuelle ou page Wikipédia.

7.3 Travail futur et améliorations proposées

Nous avons établi une liste de modifications possibles afin d'améliorer davantage notre système. Les améliorations proposées touchent l'étape de filtrage des pages Wikipédia en classes et instances, l'introduction d'un module de prétraitement permettant l'apprentissage automatique de patrons, l'introduction d'un analyseur sémantique des phrases, et finalement la désambiguïsation par des URIs de DBpedia.

En ce qui concerne le module de filtrage des pages Wikipédia et de la détermination de la nature des pages, il est possible d'ajouter des facteurs pour augmenter la précision du filtrage. En effet,

---

[24] www.westlab.polymtl.ca/AXIOpediaWebApp/Ontology

en nous référant à d'autres triplets de DBpedia ou encore aux systèmes de catégories de Wikipédia, nous aurons plus d'information pour juger de la nature de la page traitée. Il serait aussi possible de nous référer à d'autres bases de connaissances, telles que OpenCyc [61] ou WordNet [62], dans lesquelles les types des pages sont identifiés à l'avance.

D'autre part, pour la liste des patrons syntaxiques, il serait avantageux d'automatiser la détection des patrons de la langue en utilisant des méthodes d'apprentissage machine sur un plus grand sous-ensemble de définitions extraites de Wikipédia. Ceci nous permettrait d'identifier des patrons moins fréquents, ou des patrons plus complexes et plus difficiles à détecter. Une fois la liste de patrons établie, nous pourrons les annoter manuellement afin de définir quelles sont les axiomes à créer lors de la détection de leurs occurrences.

Un autre avantage serait l'analyse sémantique des phrases. Nous avons noté dans l'article 3 des points que nous ne traitons pas. Tout d'abord les références aux pronoms dans les axiomes causent un problème qu'on pourrait éviter en effectuant une désambiguïsation des pronoms. Mais encore, l'ajout d'un module qui faisait la distinction entre les trois types d'adjectif mentionné dans l'article 3, (les adjectifs subjectifs, intersectifs et privatifs), représenterait un grand pas quant à la qualité des axiomes générés.

Finalement, en ce qui concerne la désambiguïsation d'URI, nous nous basons pour l'instant sur une simple comparaison de chaines de caractères. Afin de mieux intégrer notre base de données avec celle de DBpedia, il serait nécessaire d'améliorer cette étape en utilisant, entre autres, les méthodes de désambiguïsation proposées dans notre article de la compétition *Open Knowledge Extraction*.

CHAPITRE 8 CONCLUSION

Ce mémoire présente notre approche pour l'extraction d'axiomes et de relations taxonomiques et d'instances à partir de définitions textuelles. Nous présentons un système complet, qui prend en entrée une définition textuelle, et selon le type de l'entité définie, génère un axiome si l'entité est une classe, ou un type si l'entité est une instance. Nous utilisons SPARQL et RDF pour représenter la structure grammaticale de la phrase et y rechercher des occurrences de patrons syntaxiques que nous alignons à des triplets OWL à créer. Nos évaluations reflètent des valeurs de précision et rappel élevées, mais avec un faible accord entre annotateurs pour certaines évaluations, ce qui pourrait être réglé par des évaluations sur un plus grand ensemble de données et un plus grand nombre d'annotateurs.

Ce travail de recherche contribue à l'amélioration du Web sémantique et aborde un sujet peu exploité dans le domaine, soit, l'extraction d'axiomes et de règles logiques définissant les classes. De plus, nous prenons en considération le typage des instances que nous ajoutons à la base de connaissance que nous construisons. L'étape de désambiguïsation permet de relier les ressources de la base de connaissance à ceux de DBpedia; nous créons ainsi une base de connaissance, contenant une ontologie la plus complète possible, tout en étant intégrable à DBpedia.

Nous avons fourni un effort particulier pour la création de l'ontologie AXIOpedia, qui englobe les définitions de l'ensemble des pages de Wikipédia, et nous offrons à la communauté du Web sémantique et aux chercheurs du domaine, la possibilité de consulter notre ontologie et d'utiliser notre outil à partir d'un service Web que nous avons mis en place.

BIBLIOGRAPHIE

[1] H. Paulheim et C. Bizer, «Improving the Quality of Linked Data Using Statistical Distributions,» *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 10, pp. 63-86, 2014.

[2] T.-B. Lee, «Semantic Web and Linked Data,» [En ligne]. Available: https://www.w3.org/2009/Talks/0120-campus-party-tbl/#(1). [Accès le 15 February 2017].

[3] W3C, «RDF 1.1 Semantics,» 25 February 2014. [En ligne]. Available: https://www.w3.org/TR/rdf11-mt/. [Accès le 25 February 2017].

[4] W3C, «RDF Schema 1.1,» 25 February 2014. [En ligne]. Available: https://www.w3.org/TR/rdf-schema/. [Accès le 15 February 2017].

[5] W3C, «OWL Web Ontology Language,» 10 February 2004. [En ligne]. Available: https://www.w3.org/TR/owl-features/. [Accès le 15 February 2017].

[6] W3C, «SPARQL Query Language for RDF,» 15 January 2008. [En ligne]. Available: https://www.w3.org/TR/rdf-sparql-query/. [Accès le 15 February 2017].

[7] F. M. Suchanek, G. Kasneci et G. Weikum, «Yago: A large ontology from wikipedia and wordnet,» *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, n° %13, pp. 203-217, 2008.

[8] K. Nakayama, M. Pei, M. Erdmann, M. Ito, M. Shirakawa, T. Hara et S. Nishio, «Wikipedia Mining Wikipedia as a Corpus for Knowledge Extraction,» 2008.

[9] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak et Z. Ives, «Dbpedia: A nucleus for a web of open data,» chez *In The semantic web*, Berlin Heidelber, Springer, 2007, pp. 722-735.

[10] A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead et S. Soderland, «Textrunner: open information extraction on the web,» chez *Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, . Association for Computational Linguistics.*, 2007.

[11] F. Wu et D. S. Weld, «Open information extraction using Wikipedia,» chez *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics.*, 2010.

[12] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak et S. Hellmann, «DBpedia Crystalization Point for the Web of Data,» *Web Semantics: science, services and agents on the world wide web*, vol. 7, n° %13, pp. 154-165, 2009.

[13] A. Zaveri, D. Kontokostas, M. A. Sherif, L. Bühmann, M. Morsey, S. Auer et J. Lehmann, «User Driven Quality Evaluation of DBpedia,» chez *Proceedings of the 9th International Conference on Semantic Systems. ACM.*, 2013, 97-104.

[14] P. Buitelaar, P. Cimiano et B. Magnini, «Ontology learning from text: An overview,» chez *Ontology learning from text: Methods, evaluation and applications*, IOS Press, 2005, pp. 3-12.

[15] K. T. Frantzi et S. Ananiadou, «The C-value/NC-value domain-independent method for multi-word term extraction,» *Journal of Natural Language Processing*, vol. 6, n° %13, pp. 145-179, 1999.

[16] P. Pantel et D. Lin, «A statistical corpus-based term extractor,» chez *Conference of the Canadian Society for Computational Studies of Intelligence. Springer.*, Berlin Heidelberg, 2001.

[17] V. I. Spitkovsky et A. X. Chang, «A Cross-Lingual Dictionary for English Wikipedia Concepts,» *LREC*, pp. 3168-3175, 2012.

[18] D. Nguyen, A. Overwijk, C. Hauff, D. R. Trieschnigg, D. Hiemstra et F. De Jong, «WikiTranslate: query translation for cross-lingual information retrieval using only Wikipedia." In,» chez *Workshop of the Cross-Language Evaluation Forum. Springer.*, Berlin Heidelberg, 2008.

[19] D. Yarowsky, «Word-sense disambiguation using statistical models of Roget's categories trained on large corpora,» chez *Proceedings of the 14th conference on Computational linguistics-Volume 2. Association for Computational Linguistics.*, 1992.

[20] D. Yarowsky, «Unsupervised word sense disambiguation rivaling supervised methods,» chez *Proceedings of the 33rd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics.*, 1995.

[21] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld et A. Yates, «Web-scale information extraction in knowitall:(preliminary results),» chez *Proceedings of the 13th international conference on World Wide Web. ACM.*, 2004.

[22] R. Evans et S. Street, «A framework for named entity recognition in the open domain,» *Recent Advances in Natural Language Processing III: Selected Papers from RANLP*, vol. 260, n° %1110, pp. 267-274, 2003.

[23] S. P. Ponzetto et M. Strube, «Wikitaxonomy : A Large Scale Knowledge Resource,» *ECAI*, vol. 178, pp. 751-752, 2008.

[24] M. Röder, R. Usbeck, R. Speck et A.-C. Ngonga Ngomo, «CETUS – A Baseline Approach to Type Extraction,» *Springer: Communications in Computer and Information Science*, vol. 548, pp. 16-27, 2016.

[25] S. Faralli et S. P. Ponzetto, «DWS at the 2016 Open Knowledge Extraction Challenge: A Hearst-Like Pattern-Based Approach to Hypernym Extraction and Class Induction,» *Springer : Communications in Computer and Information Science*, vol. 641, pp. 48-60, 2016.

[26] J. Gao et S. Mazumda, «Exploiting Linked Open Data to Uncover Entity Types,» *Springer : Communications in Computer and Information Science,* vol. 548, pp. 51-62, 2016.

[27] S. Consoli et D. Reforgiato Recupero, «Using FRED for Named Entity Resolution, Linking and Typing for Knowledge Base Population,» *Springer : Communications in Computer and Information Science,* vol. 548, pp. 40-50, 2016.

[28] T. Kliegr et O. Zamazal, «LHD 2.0: A text mining approach to typing entities in knowledge graphs,» *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 39, pp. 47-61, 2016.

[29] Q. Liu, K. Xu, L. Zhang, H. Wang, Y. Yu et Y. Pan, «Catriple: Extracting triples from wikipedia categories,» chez *Asian Semantic Web Conference. Springer.*, Berlin Heidelberg, 2008.

[30] G. Wang, Y. Yu et H. Zhu, «Pore: Positive-only relation extraction from wikipedia text,» chez *The Semantic Web. Springer.*, Berlin Heidelberg, 2007.

[31] B. Lorenz et J. Lehman, «Pattern Base Knowledge Base Enrichment,» *International Semantic Web Conference - Springer Berlin Heidelberg,* pp. 33-48, 2013.

[32] J. Völker, P. Hitzler et P. Cimiano, «Acquisition of OWL DL Axioms from Lexical Resources,» *Springer : Lecture Notes in Computer Science,* vol. 4519, pp. 670-685, 2007.

[33] J. Völker et M. Niepert, «Statistical Schema Induction,» *Extended Semantic Web Conference - Springer Berlin Heidelberg,* 2011.

[34] G. Töpper, M. Knuth et H. Sack, «DBpedia ontology enrichment for inconsistency detection,» chez *Proceedings of the 8th International Conference on Semantic Systems. ACM.*, 2012.

[35] M. Morsey, J. Lehmann, S. Auer, C. Stadler et S. Hellmann, «Dbpedia and the live extraction of structured data from wikipedia,» *Program,* vol. 46, n° %12, pp. 157-181, 2012.

[36] Z. Cäcilia, V. Nastase et M. Strube, «Distinguishing between Instances and Classes in the Wikipedia Taxonomy,» *European Semantic Web Conference - Springer Berlin Heidelberg*, pp. 376-387, 2008.

[37] L. Haidar-Ahmad, L. Font, A. Zouaq et M. Gagnon, «Entity typing and linking using sparql patterns and DBpedia.,» *Semantic Web Evaluation Challenge. Springer International Publishing.*, vol. 641, pp. 61-75, 2016.

[38] H. Paulheim et C. Bizer, «Type inference on noisy rdf data,» chez *International Semantic Web Conference - Springer.*, Berlin Heidelberg, 2013.

[39] P. N. Mendes, M. Jakob, A. García-Silva et C. Bizer, «DBpedia spotlight: shedding light on the web of documents,» chez *Proceedings of the 7th international conference on semantic systems. ACM.*, 2011.

[40] H. Xianpei et J. Zhao, «Named entity disambiguation by leveraging wikipedia semantic knowledge,» chez *Proceedings of the 18th ACM conference on Information and knowledge management. ACM*, 2009.

[41] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater et G. Weikum, «Robust disambiguation of named entities in text,» chez *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011.

[42] C. Silviu, «Large-scale named entity disambiguation based on Wikipedia data,» Redmond, WA, 2007.

[43] M.-C. De Marneffe et C. D. Manning, «The Stanford typed dependencies representation,» chez *Proc. of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation. ACL.*, 2008.

[44] L. Font, A. Zouaq et M. Gagnon, «Assessing the Quality of Domain Concepts Descriptions in DBpedia,» pp. 254-261, 2015.

[45] L. Bühmann, D. Fleischhacker, J. Lehmann, A. Melo et J. Völker, «Inductive Lexical Learning of Class Expressions,» *Springer International Publishing. In Knowledge Engineering and Knowledge Management.*, pp. 42-53, 2014.

[46] J. Völker, Learning expressive ontologies, IOS Press, 2009.

[47] «OKE 2015 Challenge Description.,» [En ligne]. Available: https://github.com/anuzzolese/oke-challenge/blob/master/participating%20systems/OKE2015_challengeDescription.pdf. [Accès le 20 March 2016].

[48] D. Nadeau et S. Sekine, «A survey of named entity recognition and classification,» *Lingvisticae Investigationes*, vol. 30, n° %11, pp. 3-26, 2007.

[49] M. A. Hearst, «Automatic acquisition of hyponyms from large text corpora,» *Proceedings of the 14th conference on Computational linguistics - Association for Computational Linguistic*, vol. 2, pp. 539-545, 1992.

[50] B. Min, S. Shi, R. Grishman et C.-Y. Lin, «Ensemble semantics for large-scale unsupervised relation extraction,» chez *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.

[51] M. Mintz, S. Bills, R. Snow et D. Jurafsky, «Distant supervision for relation extraction without labeled data,» chez *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics*, 2009.

[52] L. Otero-Cerdeira, F. J. Rodríguez-Martínez et A. Gómez-Rodríguez, «Ontology matching: A literature review,» *Expert Systems with Applications*, vol. 42, n° %12, p. 949 – 971, 2015.

[53] C. Bizer, T. Heath, D. Ayers et Y. Raimond, «Interlinking open data on the web,» chez *Demonstrations track, 4th european semantic web conference*, Innsbruck, Austria, 2007.

[54] F. Giunchiglia, P. Shvaiko et M. Yatskevich, «Discovering missing background knowledge in ontology matching,» *ECAI,* vol. 141, pp. 382-386, 2006.

[55] M. Kachroudi, E. Ben Moussa, S. Zghal et S. Ben Yahia, «Ldoa results for oaei 2011,» *In Proceedings of the 6th International Conference on Ontology Matching,* vol. 814, pp. 148-155, 2011.

[56] B. Johan, «Wide-Coverage Semantic Analysis with Boxer,» chez *Proceedings of the 2008 Conference on Semantics in Text Processing. Association for Computational Linguistics.*, 2008.

[57] J. Gao et S. Mazumdar, «Exploiting linked open data to uncover entity type,» *Springer International Publishing-Semantic Web Evaluation Challenge.,* pp. 51-62, 31 May 2015.

[58] «Ontology Design Patterns,» [En ligne]. Available: http://ontologydesignpatterns.org/ont/dbpedia_2014_imports.owl. [Accès le 20 March 2014].

[59] «CETUS (2015). DolCE_YAGO_mapping,» [En ligne]. Available: http://github.com/AKSW/Cetus/blob/master/DOLCE_YAGO_links.nt. [Accès le 20 March 2016].

[60] L. Font, A. Zouaq et M. Gagnon, «Assessing the quality of domain concepts descriptions in DBpedia,» chez *Signal-Image Technology & Internet-Based Systems (SITIS). 11th International Conference on. IEEE.*, 2015.

[61] R. Usbeck, M. Röder, A.-C. Ngonga Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann et P. Ferragina, «GERBIL: general entity annotator benchmarking framework.,» chez *Proceedings of the 24th International Conference on World Wide Web*.

[62] L. Bühmann et J. Lehmann, «Pattern Based Knowledge Base Enrichment,» chez *International Semantic Web Conference. Springer*, Berlin Heidelberg, 2013.

[63] L. Font, A. Zouaq et M. Gagnon, «Assessing the quality of domain concepts descriptions in DBpedia,» chez *Signal-Image Technology & Internet-Based Systems (SITIS). 11th International Conference on. IEEE.*, 2015.

[64] W. Wong, W. Liu et M. Bennamoun, «Ontology learning from text: A look back and into the future,» *ACM Computing Surveys (CSUR),* vol. 44, n° %14, p. 20, 2012.

[65] L. Haidar-Ahmad, A. Zouaq et M. Gagnon, «Automatic Extraction of Axioms from Wikipedia Using SPARQL,» *Springer Link: Lecture Notes in Computer Science,* vol. 9989, pp. 60-64, 2016.

[66] L. Haidar-Ahmad, L. Font, A. Zouaq et M. Gagnon, «Entity typing and linking using sparql patterns and DBpedia.,» *Semantic Web Evaluation Challenge. Springer International Publishing.*, vol. 641, pp. 61-75, 2016.

[67] M. Sabou, E. Blomqvist, T. Di Noia, H. Sack et T. Pellegrini, «User Driven Quality Evaluation of DBpedia,» chez *Proceedings of the 9th International Conference on Semantic Systems*, Austria, 2013.

[68] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak et S. Hellmann, «DBpedia Crystalization Point for the Web of Data,» *Web Semantics: science, services and agents on the world wide web 7.3,* vol. 7, n° %13, pp. 154-165, 2009.

[69] F. Orlandi et A. Passant, «Modelling provenance of DBpedia resources using Wikipedia contributions,» *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 9, n° %12, pp. 149-164, 2011.

[70] G. Töpper, M. Knuth et H. Sack, «DBpedia ontology enrichment for inconsistency detection,» chez *Proceedings of the 8th International Conference on Semantic Systems. ACM.*, 2012.

[71] M. Morsey, J. Lehmann, S. Auer, C. Stadler et S. Hellmann, «Dbpedia and the live extraction of structured data from wikipedia,» *Program,* vol. 46, n° %12, pp. 157-181, 2012.

[72] J. Waitelonis, N. Ludwig, M. Knuth et H. Sack, «Whoknows? evaluating linked data heuristics with a quiz that cleans up dbpedia,» *nteractive Technology and Smart Education,* vol. 8, n° %14, pp. 236-248, 2011.

[73] K. Nakayama, M. Pei, M. Erdmann, M. Ito, M. Shirakawa, T. Hara et S. Nishio, «Wikipedia Mining Wikipedia as a Corpus for Knowledge Extraction,» 2008.

[74] O. Medelyan, D. Milne, C. Legg et I. H. Witten, «Mining meaning from Wikipedia,» *International Journal of Human-Computer Studies,* vol. 67, n° %19, pp. 716-754, 2009.

[75] P. Buitelaar, P. Cimiano et B. Magnini, «Ontology learning from text: An overview,» chez *Ontology learning from text: Methods, evaluation and applications,* IOS Press, 2005, pp. 3-12.

[76] S. P. Ponzetto et M. Strube, «Wikitaxonomy : A Large Scale Knowledge Resource,» *ECAI,* vol. 178, pp. 751-752, 2008.

[77] M. Röder, R. Usbeck, R. Speck et A.-C. Ngonga Ngomo, «CETUS – A Baseline Approach to Type Extraction,» *Springer: Communications in Computer and Information Science,* vol. 548, pp. 16-27, 2016.

[78] J. Gao et S. Mazumda, «Exploiting Linked Open Data to Uncover Entity Types,» *Springer : Communications in Computer and Information Science,* vol. 548, pp. 51-62, 2016.

[79] S. Consoli et D. Reforgiato Recupero, «Using FRED for Named Entity Resolution, Linking and Typing for Knowledge Base Population,» *Springer : Communications in Computer and Information Science,* vol. 548, pp. 40-50, 2016.

[80] S. Faralli et S. P. Ponzetto, «DWS at the 2016 Open Knowledge Extraction Challenge: A Hearst-Like Pattern-Based Approach to Hypernym Extraction and Class Induction,» *Springer : Communications in Computer and Information Science,* vol. 641, pp. 48-60, 2016.

[81] J. Bos, «Wide-coverage semantic analysis with boxer,» *Proceedings of the 2008 Conference on Semantics in Text Processing - Association for Computational Linguistics,* pp. 277-286, 2008.

[82] M. A. Hearst, «Automatic acquisition of hyponyms from large text corpora,» *Proceedings of the 14th conference on Computational linguistics - Association for Computational Linguistic,* vol. 2, pp. 539-545, 1992.

[83] T. Kliegr et O. Zamazal, «LHD 2.0: A text mining approach to typing entities in knowledge graphs,» *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 39, pp. 47-61, 2016.

[84] Q. Liu, K. Xu, L. Zhang, H. Wang, Y. Yu et Y. Pan, «Catriple: Extracting triples from wikipedia categories,» chez *Asian Semantic Web Conference. Springer.*, Berlin Heidelberg, 2008.

[85] F. Wu et D. S. Weld, «Open information extraction using Wikipedia,» chez *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics.*, 2010.

[86] A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead et S. Soderland, «Textrunner: open information extraction on the web,» chez *Human Language Technologies: The Annual*

Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, . Association for Computational Linguistics., 2007.

[87] J. Völker, P. Hitzler et P. Cimiano, «Acquisition of OWL DL Axioms from Lexical Resources,» *Springer : Lecture Notes in Computer Science,* vol. 4519, pp. 670-685, 2007.

[88] J. Völker et M. Niepert, «Statistical Schema Induction,» *Extended Semantic Web Conference - Springer Berlin Heidelberg,* 2011.

[89] Z. Cäcilia, V. Nastase et M. Strube, «Distinguishing between Instances and Classes in the Wikipedia Taxonomy,» *European Semantic Web Conference - Springer Berlin Heidelberg,* pp. 376-387, 2008.

[90] J. Rose Finkel, T. Grenager et C. Manning, «Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling,» chez *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, 2005.

[91] M.-C. De Marneffe et C. D. Manning, «The Stanford typed dependencies representation,» chez *Proc. of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation. ACL.*, 2008.

[92] I. Terziev, A. Kiryakov et D. Manov., *Base Upper-Level Ontology (BULO) guidance. SEKT.*, Ontotext Lab, Sirma AI EAD (Ltd.), 2004.

[93] J. Cohen, «A coefficient of agreement for nominal scales,» *Educational and psychological measurement,* vol. 20, n° %11, pp. 37-46, 1960.

[94] Cycorp., «OpenCyc,» [En ligne]. Available: http://opencyc.org/. [Accès le 15 February 2017].

[95] G. A. Miller, «WordNet: a lexical database for English,» *Communications of the ACM,* vol. 38, n° %111, pp. 39-41, 1995.

[96] Cycorp., «OpenCyc,» [En ligne]. Available: http://opencyc.org/. [Accès le 15 February 2017].

[97] G. A. Miller, «WordNet: a lexical database for English,» *Communications of the ACM,* vol. 38, n° %111, pp. 39-41, 1995.

[98] B. Johan, «Wide-coverage semantic analysis with boxer,» *Proceedings of the 2008 Conference on Semantics in Text Processing - Association for Computational Linguistics,* pp. 277-286, 2008.

[99] G. Salton et C. Buckley, «Term-weighting approaches in automatic text retrieval,» *Information processing & management,* vol. 24, n° %15, pp. 513-523, 1988.

[100] W. Wong, W. Liu et M. Bennamoun, «Ontology learning from text: A look back and into the future,» *ACM Computing Surveys (CSUR),* vol. 44, n° %14, p. 20, 2012.

[101] G. Wang, Y. Yu et H. Zhu, «Pore: Positive-only relation extraction from wikipedia text,» chez *The Semantic Web. Springer.*, Berlin Heidelberg, 2007.

[102] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak et Z. Ives, «Dbpedia: A nucleus for a web of open data,» chez *In The semantic web*, Berlin Heidelber, Springer, 2007, pp. 722-735.

[103] F. M. Suchanek, G. Kasneci et G. Weikum, «Yago: A large ontology from wikipedia and wordnet,» *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 6, n° %13, pp. 203-217, 2008.

[104] G. Salton et C. Buckley, «Term-weighting approaches in automatic text retrieval,» *Information processing & management,* vol. 24, n° %15, pp. 513-523, 1988.

[105] K. T. Frantzi et S. Ananiadou, «The C-value/NC-value domain-independent method for multi-word term extraction,» *Journal of Natural Language Processing,* vol. 6, n° %13, pp. 145-179, 1999.

[106] P. Pantel et D. Lin, «A statistical corpus-based term extractor,» chez *Conference of the Canadian Society for Computational Studies of Intelligence. Springer.*, Berlin Heidelberg, 2001.

[107] V. I. Spitkovsky et A. X. Chang, «A Cross-Lingual Dictionary for English Wikipedia Concepts,» *LREC*, pp. 3168-3175, 2012.

[108] D. Nguyen, A. Overwijk, C. Hauff, D. R. Trieschnigg, D. Hiemstra et F. De Jong, «WikiTranslate: query translation for cross-lingual information retrieval using only Wikipedia,» chez *Workshop of the Cross-Language Evaluation Forum. Springer.*, Berlin Heidelberg, 2008.

[109] D. Yarowsky, «Unsupervised word sense disambiguation rivaling supervised methods,» chez *Proceedings of the 33rd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics.*, 1995.

[110] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld et A. Yates, «Web-scale information extraction in knowitall:(preliminary results),» chez *Proceedings of the 13th international conference on World Wide Web. ACM.*, 2004.

[111] R. Evans et S. Street, «A framework for named entity recognition in the open domain,» *Recent Advances in Natural Language Processing III: Selected Papers from RANLP*, vol. 260, n° %1110, pp. 267-274, 2003.

[112] L. Bühmann et J. Lehmann, «Pattern Based Knowledge Base Enrichment,» *International Semantic Web Conference - Springer Berlin Heidelberg*, pp. 33-48, 2013.