UNIVERSITE DE PAU ET DES PAYS DE L'ADOUR

ECOLE DOCTORALE DES SCIENCES EXACTES ET LEURS APPLICATIONS

Towards RDF Normalization

Prepared by

Regina Paola TICONA HERRERA

A thesis submitted in fulfillment for the degree of Doctor of Phylosofy in Computer Science

06 July 2016

To the memory of my grandmother Eliana NIETO ESPEJO.

Acronyms

CERN	*The European Organization for Nuclear Research*
HITS	*Hypertext-Induced Topic Search*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IETF	*Internet Engineering Task Force*
IRI	*Internationalized Resource Identifier*
LOD	*Linking Open Data*
MCF	*Meta-Content Framework*
N3	*Notation 3*
OWL	*Web Ontology Language*
RDF	*Resource Description Framework*
RDF(S)	*RDF Schema*
SW	*Semantic Web*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
URN	*Uniform Resource Name*
XML	*eXtensible Markup Language*
W3C	*World Wide Web Consortium*
WWW	*World Wide Web*

Acknowledgments

I would like to express my sincere gratitude and humble acknowledgement to all the people who have contributed to the preparation and completion of this research.

First of all, I would like to thank to my professors and academic supervisors Dr. Richard Chbeir of the T2I Laboratory, University of Pau, as well as Dr. Joe Tekli of Lebanese American University, and Dr. Sébastien Laborie of the T2I Laboratory, University of Pau whose successful teachings have made this dissertation possible. I also thank the University of Pau for giving me the opportunity to undertake this PhD.

I would like to express my mere expression of gratitude to Dr. Richard Chbeir, for his confidence to accept me as his student and for his close supervision, exigency, and constant presence during the past four years. My humble gratitude goes to him for his big patience, support, comments, teachings, help and generous guidance during the completion of my PhD and for his time spent in reviewing and improving this work. His support and guidance were key players in making this research feasible. Equally, I would like to thank my co-supervisors: Dr. Joe Tekli for his invaluable guidance, support in all the aspects, comprehension, encouragement, teachings, trust, help, valuable advices, and commitment in improving always the work and the time spent in reviewing the report, and Dr. Sébastien Laborie for his guidance, motivation, encouragement, advice, and the time spent in reviewing the work during the thesis. Without all of them I could not have got this far, it is a pleasure to know them and a privilege to work all this time with them.

I would like to show my gratitude to Dr. Germán Chávez Contreras, Rector of the Universidad Católica San Pablo from Peru, for his support to fulfill this PhD. Also, I want to thank: i) Raquel Patiño Escarcina for pushing me to do the PhD and sending my application without her encouragement probably I would not be here, ii) Ernesto Cuadros Vargas for introducing me in the research world and giving me the chance to meet Richard Chbeir, and iii) Alejandro Estenós and Gonzalo Fernández for giving me their unconditional support.

I am thankful for my friends that more than friends were my family in Europe: i) Marita Vargas, Antonio Flores, Analía Boggia, Irvin Dongo, Renato Guzmán, Denisse Muñante, Aritz

Echenique, Aizea Lojo, Joseba Nevado, and Marta Toribio, and ii) my friends in France: Elie Raad, Eliana Raad, Fernanda Sarubbi, Rana Mn, Ghada Ben-Nejma, Keling Da, Solomon Asres, Chinnapong Angsuchotmetee (Dia), Riadh Karchoud and all the members of the lab. Above all, I am thankful for their advices, encouragement and patience to me. I am also grateful to all my friends that for space limitations I will not mention name by name, but I will have always all the beautiful memories with them in my heart.

I would like to show my gratitude to my family for their support and encouragement, especially to my father for the opportunities he has given me in life.

I also want to thanks to all the institutions/organizations that supported me in different ways as FINCYT (Peru), Campus France, LIUPPA (France), and UCSP (Peru).

Lastly and most importantly, my deepest gratitude and love goes to God who filled me with strength, patience, perseverance, and wisdom to finish this work.

Resumé

Depuis ces dernières décennies, des millions d'internautes produisent et échangent des données sur le Web. Ces informations peuvent être structurées, semi-structurées et/ou non-structurées, tels que les blogs, les commentaires, les pages Web, les contenus multimédias, etc. Afin de faciliter la publication ainsi que l'échange de données, le World Wide Web Consortium (ou W3C) a défini en 1999 le standard RDF. Ce standard est un modèle qui permet notamment de structurer une information sous la forme d'un réseau de données dans lequel il est possible d'y attacher des descriptions sémantiques. Ce modèle permet donc d'améliorer l'interopérabilité entre différentes applications exploitant des données diverses et variées présentes sur le Web.

Actuellement, une grande quantité de descriptions RDF est disponible en ligne, notamment grâce à des projets de recherche qui traitent du Web de données liées, comme par exemple DBpedia et LinkedGeoData. De plus, de nombreux fournisseurs de données ont adopté les technologies issus de cette communauté du Web de données en partageant, connectant, enrichissant et publiant leurs informations à l'aide du standard RDF, comme les gouvernements (France, Canada, Grande-Bretagne, etc.), les universités (par exemple Open University) ainsi que les entreprises (BBC, CNN, etc.). Il en résulte que de nombreux acteurs actuels (particuliers ou organisations) produisent des quantités gigantesques de descriptions RDF qui sont échangées selon différents formats (RDF/XML, Turtle, N-Triple, etc.).

Néanmoins, ces descriptions RDF sont souvent verbeuses et peuvent également contenir de la redondance d'information. Ceci peut concerner à la fois leur structure ou bien leur sérialisation (ou le format) qui en plus souffre de multiples variations d'écritures possibles au sein d'un même format. Tous ces problèmes induisent des pertes de performance pour le stockage, le traitement ou encore le chargement de ce type de descriptions.

Dans cette thèse, nous proposons de nettoyer les descriptions RDF en éliminant les données redondantes ou inutiles. Ce processus est nommé “normalisation” de descriptions RDF et il est une étape essentielle pour de nombreuses applications, telles que la similarité entre descriptions, l’alignement, l’intégration, le traitement des versions, la classification, l’échantillonnage, etc. Pour ce faire, nous proposons une approche intitulée R2NR qui à partir de différentes descriptions relatives à une même information produise une et une seule description normalisée qui est optimisée en fonction de multiples paramètres liés à une application cible. Notre approche est illustrée en décrivant plusieurs cas d’étude (simple pour la compréhension mais aussi plus réaliste pour montrer le passage à l’échelle) nécessitant l’étape de normalisation.

La contribution de cette thèse peut être synthétisée selon les points suivants:

i) Produire une description RDF normalisée (en sortie) qui préserve les informations d’une description source (en entrée),

ii) Éliminer les redondances et optimiser l’encodage d’une description normalisée,

iii) Engendrer une description RDF optimisée en fonction d’une application cible (chargement rapide, stockage optimisée...),

iv) Définir de manière complète et formelle le processus de normalisation à l’aide de fonctions, d’opérateurs, de règles et de propriétés bien fondées, etc.

v) Fournir un prototype RDF2NormRDF (avec deux versions : en ligne et hors ligne) permettant de tester et de valider l’efficacité de notre approche.

Afin de valider notre proposition, le prototype RDF2NormRDF a été utilisé avec une batterie de tests. Nos résultats expérimentaux ont montré des mesures très encourageantes par rapport aux approches existantes, notamment vis-à-vis du temps de chargement ou bien du stockage d’une description normalisée, tout en préservant le maximum d’informations.

Abstract

Over the past three decades, millions of people have been producing and sharing information on the Web, this information can be structured, semi-structured, and/or non-structured such as blogs, comments, Web pages, and multimedia data, etc., which require a formal description to help their publication and/or exchange on the Web. To help address this problem, the Word Wide Web Consortium (or W3C) introduced in 1999 the RDF standard as a data model designed to standardize the definition and use of metadata, in order to better describe and handle data semantics, thus improving interoperability, and scalability, and promoting the deployment of new Web applications.

Currently, billions of RDF descriptions are available on the Web through the Linked Open Data cloud projects (e.g., DBpedia and LinkedGeoData). Also, several data providers have adopted the principles and practices of the Linked Data to share, connect, enrich and publish their information using the RDF standard, e.g., Governments (e.g., Canada Government), universities (e.g., Open University) and companies (e.g., BBC and CNN). As a result, both individuals and organizations are increasingly producing huge collections of RDF descriptions and exchanging them through different serialization formats (e.g., RDF/XML, Turtle, N-Triple, etc.).

However, many available RDF descriptions (i.e., graphs and serializations) are noisy in terms of structure, syntax, and semantics, and thus may present problems when exploiting them (e.g., more storage, processing time, and loading time). In this study, we propose to clean RDF descriptions of redundancies and unused information, which we consider to be an essential and required stepping stone toward performing advanced RDF processing as well as the development of RDF databases and related applications (e.g., similarity computation, mapping, alignment, integration, versioning, clustering, and classification, etc.). For that purpose, we have defined a framework entitled R2NR which normalizes different RDF descriptions pertaining to the same information into one normalized representation, which can then be tuned both at the graph level and at the serialization level, depending on the target application and user requirements. We illustrate this approach by introducing use cases (real and synthetics) that need to be normalized.

The contributions of the thesis can be summarized as follows:

i) Producing a normalized (output) RDF representation that preserves all the information in the source (input) RDF descriptions,

ii) Eliminating redundancies and disparities in the normalized RDF descriptions, both at the logical (graph) and physical (serialization) levels,

iii) Computing a RDF serialization output adapted w.r.t. the target application requirements (faster loading, better storage, etc.),

iv) Providing a mathematical formalization of the normalization process with dedicated normalization functions, operators, and rules with provable properties, and

v) Providing a prototype tool called RDF2NormRDF (desktop and online versions) in order to test and to evaluate the approach's efficiency.

In order to validate our framework, the prototype RDF2NormRDF has been tested through extensive experimentations. Experimental results are satisfactory show significant improvements over existing approaches, namely regarding loading time and file size, while preserving all the information from the original description.

Table of Contents

List of Tables

List of Figures

Chapter 1

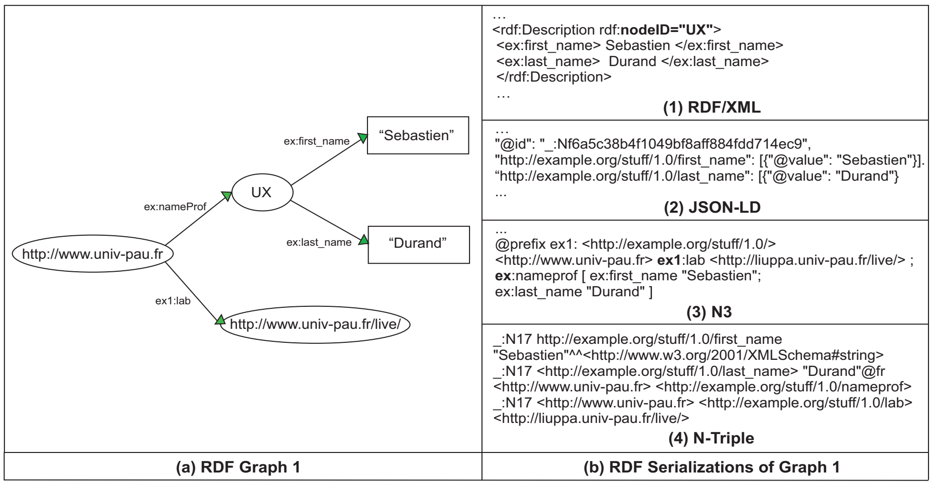
Introduction

The Semantic Web is a global information space consisting of interlinked data about resources [BL09], aiming to assign/extend data with well-defined meaning that can be understood and utilized by machines to improve the quality of the information retrieved and also perform more sophisticated data management and interchange tasks (e.g., intelligent data search, data integration, merging, classification, etc.). In this context, one of the core technologies of the Semantic Web to connect data is the *Resource Description Framework* (RDF) [MMM$^+$04, SR14], as a *World Wide Web Consortium* (W3C) standard. Basically, an RDF description is composed of a set of triples $< \textit{Subject}, \underline{\textit{Predicate}}, \textit{Object} >$, also named statements. These triples altogether form an RDF graph highlighting the semantic linkage/relationships between Web resources.

For instance, the following triple: <`http://www.univ-pau.fr`, <u>ex1:lab</u>, `http://liuppa.univ-pau.fr/live/`> states that the subject `http://www.univ-pau.fr`, identified by its Internationalized Resource Identifier (IRI), has a <u>lab</u>, identified by its own IRI: `http://liuppa.univ-pau.fr/live/`. Several RDF datasets are currently available online thanks to Linked Data [HB11] research projects[1], such as *DBpedia*, *LinkedGeoData*, *Geonames*, *New York Times*, etc. Through the initiative of *Linking Open Data* (LOD), individuals and organizations can share all their information with others based on RDF triples.

These triples are serialized to be usually stored in RDF machine readable formats such as RDF/XML [Bec04] (since XML-based technologies give more readability and provide standardized frameworks that can be used to handle such a format), N-Triple [BB01], Turtle [BBL11], N3 [BL05] or JSON-LD [SLK$^+$14]. Therefore, RDF can be described in different ways as shown in Figure 1.1 where an RDF description is represented by a graph (cf. Figure 1.1.a) and several corresponding RDF formats (serializations) (cf. Figures 1.1.b.1, 1.1.b.2, 1.1.b.3 and 1.1.b.4).

[1]`http://linkedgeodata.org`, `http://data.nytimes.com/`, `http://dbpedia.org`, `http://www.geonames.org/`, `http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/DataSets`



```
…
<rdf:Description rdf:nodeID="UX">
 <ex:first_name> Sebastien </ex:first_name>
 <ex:last_name>  Durand </ex:last_name>
 </rdf:Description>
 …
```
(1) RDF/XML

```
…
"@id": "_:Nf6a5c38b4f1049bf8aff884fdd714ec9",
"http://example.org/stuff/1.0/first_name": [{"@value": "Sebastien"}].
“http://example.org/stuff/1.0/last_name": [{"@value": "Durand"}
...
```
(2) JSON-LD

```
...
@prefix ex1: <http://example.org/stuff/1.0/>
<http://www.univ-pau.fr> ex1:lab <http://liuppa.univ-pau.fr/live/> ;
ex:nameprof [ ex:first_name "Sebastien";
ex:last_name "Durand" ]
```
(3) N3

```
_:N17 http://example.org/stuff/1.0/first_name
"Sebastien"^^<http://www.w3.org/2001/XMLSchema#string>
_:N17 <http://example.org/stuff/1.0/last_name> "Durand"@fr
<http://www.univ-pau.fr> <http://example.org/stuff/1.0/nameprof>
_:N17 <http://www.univ-pau.fr> <http://example.org/stuff/1.0/lab>
<http://liuppa.univ-pau.fr/live/>
```
(4) N-Triple

Figure 1.1: RDF description Example.

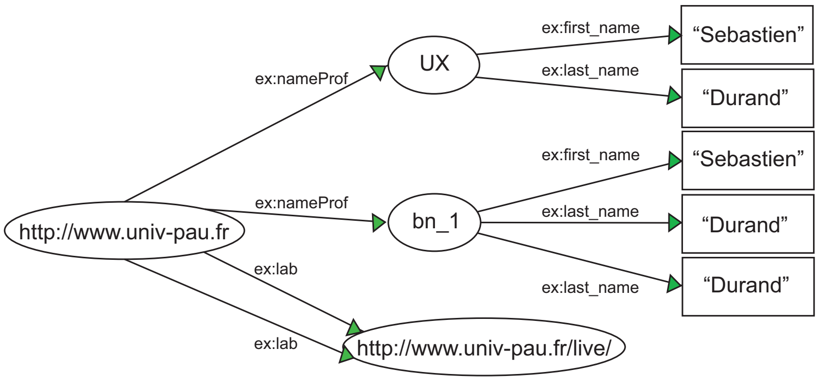


Figure 1.2: RDF Graph 2 describing the same information of Fig. 1.

In different scenarios (e.g., automatic serialization generation [STC14, Vea09], collaborative generation [Jea13], data integration [PST$^+$15], etc.), RDF descriptions might be verbose and contain several redundancies in terms of both: the structure of the graph and/or the serialization result. For instance, in automatic serialization generation, let us consider two RDF descriptions to represent the same information: Graph 1 in Figure 1.1.a and Graph 2 in Figure 1.2, which have been created by two different users. Both graphs are different in terms of structure even though they are based on (and refer to) the same information. Actually, the RDF graph in Figure 1.2 contains duplicated information (i.e., duplication of nodes and duplication of edges) that produces more statements in comparison with the RDF graph Figure 1.1.a. Additionally, even more redundancies and disparities[1] will occur when serializing the RDF graph in Figure 1.1.b, highlighting typical problems with RDF serialization (cf. Motivating examples in Sections 3.1 and 3.2):

[1]We use disparities to designate different serializations of the same information.

- The same RDF resource can be serialized in several ways (e.g., in Figure 1.1.b.1, we use the attribute value rdf:nodeID="UX" as one of several ways to represent the blank node identifier in a graph following the RDF/XML format, which can be done differently in other formats),
- The language and datatype declarations for a given RDF resource (objects of the statement) can be specified (or not) after serialization (e.g., in Figure 1.1.b.4, the datatype ***string*** is mentioned in resource ***Sebastien*** as "*Sebastien*" ^^*xsd* : *string* and the language ***fr*** is mentioned in resource ***Durand*** as "Durand"@fr, which can be omitted in other formats),
- The same URL namespace can have different short names, thus producing namespace duplications (e.g., the namespace `http://xmlns.com/foaf/1.0/` in Figure 1.1.b.1 has the short name "ex" whereas it has the short name "ex1" in Figure 1.1.b.3).

The LOD context, as we mentioned before, individuals and organizations share data for several purposes, e.g., to build new businesses, to increase online commerce, to accelerate scientific progress, etc. So, all the problems above can be duplicated, if we want to integrate/merge the information of one (or more) resource(s) using different datasets in the LOD.

For instance, the data integration of the resource Luxembourg (see Figure 1.3) provided by different datasets as DBpedia and Geonames may increase the redundancies and disparities in the merged RDF description output. Also, it has an impact in the storage, loading time, etc., e.g., RDF Graph X + RDF Graph Y + RDF Graph Z will take more space for storage and also more time for loading than the RDF graph without redundant and unused information. Considering a subgraph of the integrated RDF description output in Figure 1.4, we find more redundancies based on semantic ambiguities and IRI discrepancies[1] (cf. Motivating examples in Sections 3.3 and 3.4):

- The language and datatype declarations for a given RDF resource can change following different primitive values (e.g., string, integer, etc.) but the meaning will be the same (e.g., in Figure 1.4, the language ***es*** is mentioned in "Luxemburgo"@es and language ***en*** is mentioned "Luxembourg"@en. So, we can notice that the name of the resource changed but the meaning is the same: they simply provide the name for ***Luxembourg*** but in different languages),
- The different IRIs can refer to the same RDF resource (e.g., in Figure 1.4, the IRI `http://dbpedia.org/resource/Luxembourg` uses the DBpedia dataset and the IRI `http://sws.geonames.org/2960313/` uses the Geonames dataset, such that both IRIs identify the same resource ***Luxembourg***).

[1]We use discrepancies to designate different IRIs that refer to the same resource

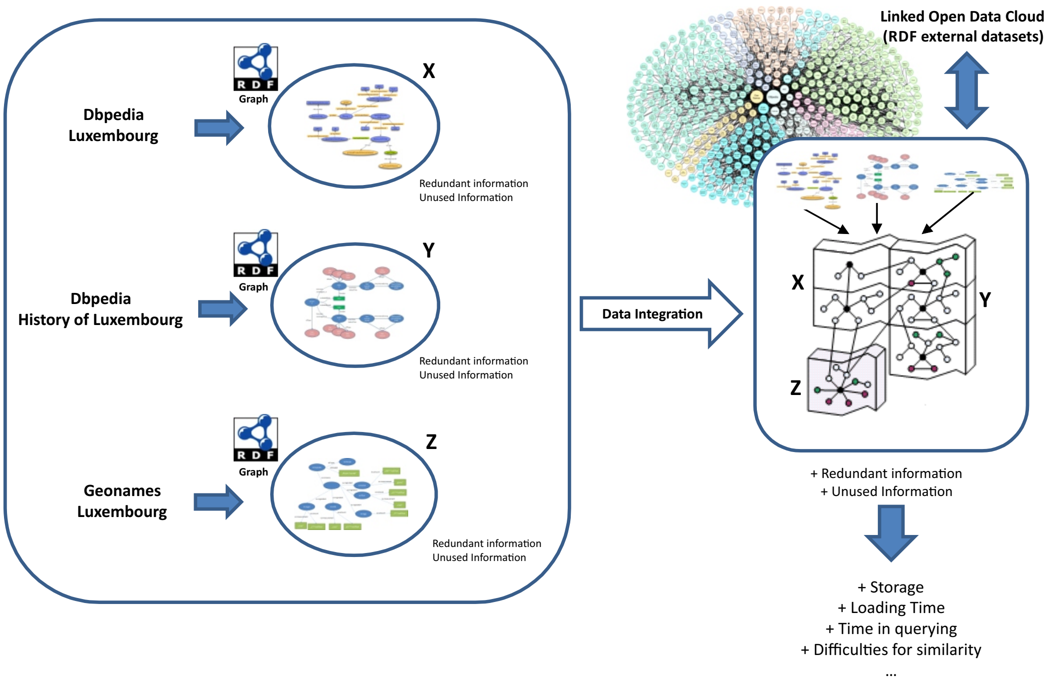


Figure 1.3: Example of Linked Data about the resource Luxembourg

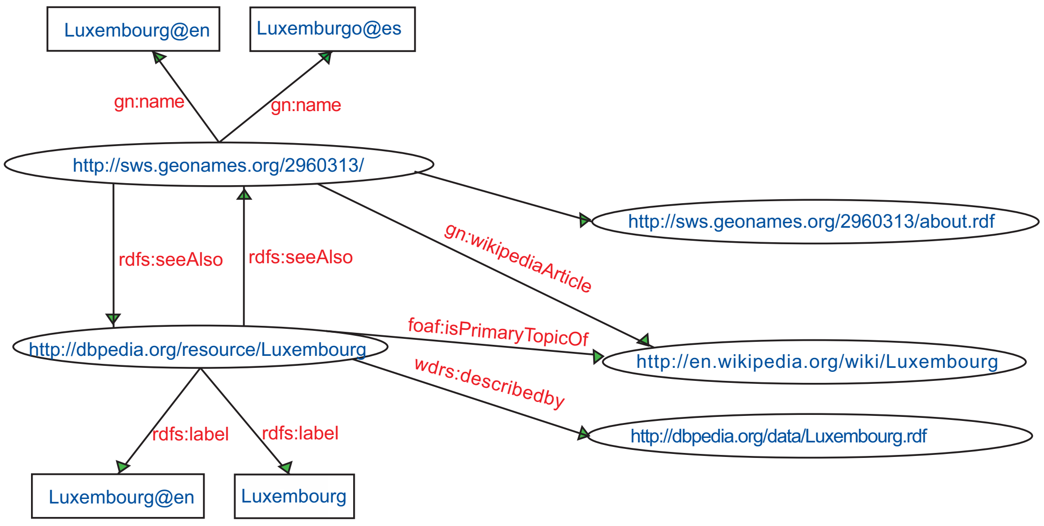

Luxembourg@en
Luxemburgo@es
gn:name
gn:name
http://sws.geonames.org/2960313/
http://sws.geonames.org/2960313/about.rdf
rdfs:seeAlso
rdfs:seeAlso
gn:wikipediaArticle
foaf:isPrimaryTopicOf
http://dbpedia.org/resource/Luxembourg
http://en.wikipedia.org/wiki/Luxembourg
wdrs:describedby
http://dbpedia.org/data/Luxembourg.rdf
rdfs:label
rdfs:label
Luxembourg@en
Luxembourg

Figure 1.4: Subgraph of the Data Integration in Figure 1.3

Such duplications and discrepancies which can occur both at the RDF graph (logical) level and/or at the serialization (physical) level, are inherently problematic in RDF processing, and would have a negative impact on the development/deployment of RDF databases [Gea11] (including storage, querying, processing time, loading time, similarity measuring, mapping, alignment, and versioning).

In this thesis, we present a proposal which provides a foundation and the main building block for full-fledged RDF normalization. We continue this chapter by first identifying the principal aim and the objectives of the thesis (Section 1.1). Next, in Section 1.2, we explain our research contributions and, in Section 1.3, we conclude this chapter with the outline of the remainder of this work.

1.1 Research Aims and Objectives

The ultimate aim of this thesis is to resolve RDF logical redundancies and physical disparities by introducing a framework for RDF normalization named R2NR, allowing to transform different RDF descriptions using the same RDF statements reference into one single (normalized) representation, while allowing to adapt RDF output serialization following application domain requirements.

Our approach targets RDF normalization through the following objectives:

1. Eliminate redundancies in RDF graphs (which is typically useful in improving graph-based RDF querying, mapping, and versioning applications) in the structure/graph (logical) level, and
2. Eliminate redundancies and disparities in the structure of RDF files in the serialization (physical) level, and adapting it to the target application domain, in order to optimize storage space and loading time.
3. Prove that our normalization process is: i) valid with respect to a set of provable properties, ii) flexible and adaptable to user and application requirements, and iii) efficient and scalable in processing large RDF repositories.

1.2 Research Contributions

Based on the aim and objectives described above, and our study of the research area (developed in Sections 4.2 and 5.2), we present the following as our primary contribution in this thesis:

1. **Syntactic RDF Normalization**

   The challenge of obtaining an RDF normalized description has only been partly addressed in the literature (cf. Section 4.2). Existing solutions do not consider all syntactic aspects of RDF including: blank node duplication, unused namespaces, etc., that we further categorize in this thesis as logical redundancies and physical disparities. In order to perform syntactic RDF normalization, we provide:

   - Systematic and complete description of R2NR's formal mathematical model, including a battery of formalized normalization rules, functions, operators, with their properties, and corresponding proofs.
   - Eliminating redundancies and disparities in the normalized RDF descriptions, both at the logical (graph) and physical (serialization) levels in the syntactic way.
   - Producing a normalized (output) RDF representation that preserves all the information in the source (input) RDF descriptions.
   - Providing several RDF serialization outputs adapted to the target application requirements (faster loading, better storage, etc.).

2. **Semantic RDF Normalization**

   In addition to considering the syntactic features of RDF, we have also extended our proposal to deal with RDF semantic normalization: considering not only the occurrence of duplicate RDF elements (i.e., repetition of identical RDF subjects/predicates/objects), but also the occurrence of semantically similar/related elements, and how these can affect normalization. As the semantic issues are inherently important in RDF descriptions, we have extended our RDF Normalization framework with a *Semantic Level* processing component, providing:

   - Mathematical formalization for the rules and functions related to solving logical redundancies based on semantic ambiguities.
   - Dedicated components to eliminate redundancies in the normalized RDF descriptions at the logical level.
   - Dedicated components to perform semantic analysis, required for solving semantic redundancies in RDF descriptions.

3. **IRI RDF Normalization**

   The second extension of our syntactic proposal of RDF normalized considers IRI discrepancies. In a new environment, such as the Semantic Web, where the IRIs are the base of all Web applications to link the information about resources in datasets of linked data, we study solutions to solve the problems of the IRI identity and IRI coreference[1]. In

[1]Coreference means that two or more IRIs are used to designate the same resource in the same way

practice, RDF descriptions can also have redundancies and disparities which we aim to solve by introducing a dedicated *IRI Level* solution, providing:

- Mathematical formalization for the rules and functions related to solving logical redundancies and physical disparities based on IRI discrepancies.
- Eliminating redundancies and disparities in the normalized RDF descriptions, both at the logical and physical levels, generated by the IRI discrepancies.
- A dedicated component to identify the IRI coreference, which is in turn required to identify and solve IRI discrepancies.

We also develop a prototype tool called *RDF2NormRDF* in two versions: Desktop and online application[1], in order to test and to evaluate our approach's: i) effectiveness: quality in detecting and eliminating redundancies and disparities in RDF descriptions, and ii) efficiency: evaluating processing time, loading time, and storage space. Our experiments target both logical and physical normalization at the syntactic level, and are being extended toward evaluating semantic normalization. As well, we present an extensive and comparative experimental evaluation study analyzing large scale experimental results in comparison with existing methods.

Results have been presented and published in the proceedings of the 34th International Conference on Conceptual Modelling ER'15 [THTC$^+$15], and the extended study is currently submitted (under review) in the International Journal on Web Semantics (Elsevier JWS) [THTCL16].

1.3 Manuscript Structure

Next, we present an overview of each of the following chapters in this thesis:

Chapter 2 (The Semantic Web: RDF and Linked Data) presents the necessary background information regarding the concepts and principles about WWW, Semantic Web, IRIs, RDF and the Web of Linked Data considered to better understanding the normalization process.

Chapter 3 (Motivating Examples) presents motivating examples, highlighting different normalization features left unaddressed by most existing approaches. Regarding these features, this chapter also presents our challenges for the RDF normalization.

Chapter 4 (Syntactic RDF Normalization) describes our syntactic proposal for RDF Normalization. This chapter also includes preliminary notions, basic definitions related

[1] `http://sigappfr.org/spider/research-projects/towards-rdf-normalization/`

to RDF logical and physical descriptions, a set of normalization functions, operators and properties, and a comparison with related works in RDF standardization and normalization. Finally, we detail our overall R2NR framework architecture and components.

Chapter 5 (Semantic and IRI RDF Normalization) describes an extension of R2NR framework architecture considering RDF Semantic and IRI problems. This chapter also includes preliminary notions related to semantic ambiguity, IRI identity, IRI coreference, and RDF semantic normalization, as well as a set of normalization functions, operators, and properties linked to solving logical redundancies and physical disparities caused by the presence of semantic ambiguities and IRI discrepancies. This extension adds two levels to our original framework: Semantic level and IRI level. In this chapter, we also present our prototype and the experimental environment describing all our datasets.

Chapter 6 (Experimental Evaluation) illustrates and discusses experimental results of evaluating the R2NR proposal that we presented in the preceding chapters, and shows results of the validation of RDF Normalization output through the fulfilment of RDF normalization properties, as well as the cost in processing time and the gain in loading time and storage space achieved after normalization.

Chapter 7 (Conclusions and Future Works) concludes our work, recapitulating our contributions and highlighting future research directions.

Chapter 2

The Semantic Web: RDF and Linked Data

"I made some electronic gadgets to control the trains. Then I ended up getting more interested in electronics than trains. Later on, when I was in college I made a computer out of an old television set."

— Tim Berners-Lee

In this chapter, we present the Semantic Web concept and its associated elements. It is important that we first understand the nature, purpose and principles of the Semantic Web before the challenges of the RDF normalization. The nature of the *Semantic Web* (SW) is linked to two motivations: 1) The first one, it is the distributed modelling of the world that allows "anyone to say anything about anything" in a globally unambiguous, machine-readable format with a shared data model, and 2) The second one, it is the infrastructure where data and schemas can be published, found and used by anyone. So, the main question for researchers was: how to publish information about resources in a way that allows interested users and software applications to find and interpret them [AV08, SCV11, Boo03]. To that end, for answering this question, we begin this chapter with a brief history of the *World Wide Web* (WWW) and the SW (Section 2.1) and a short background description related to: i) *Internationalized Resource Identifiers* (IRIs) (Section 2.2), and ii) *Resource Description Framework* (RDF) (Section 2.3). Finally, we present the relation between the Linked Data movement (Section 2.4) and the Semantic Web.

2.1 World Wide Web and Semantic Web

The WWW, named as *Web*, was invented by Tim Berners-Lee in 1989, as a collaboration tool for the High-Energy Physics research community at *The European Organization for Nuclear Research* (CERN)[1][BL89]. Berners-Lee had developed a concept for the Web as "universal information space", in his original proposal he said "We should work towards a universal linked information system, in which generality and portability are more important than fancy graphics and complex extra facilities" [BL89]. This concept was related with the main goal of Berners-Lee's proposal that was to connect the tremendous amounts of data of CERN.

Also, Tim Berners-Lee, Roy Fielding, Dan Connolly, and others were participants of the *Internet Engineering Task Force* (IETF) to create software to run the Internet [Wal01] and the result of this collaboration, around 1990, was the development of basic protocols and data formats as URIs that is a unique address used to identify each resource on the Web (see more details in Section 2.2), *HyperText Markup Language* (HTML) that is the markup language of the Web and *Hypertext Transfer Protocol* (HTTP) that allow us the retrieval of linked resources from across the Web. These three fundamental technologies remain the foundation of today's Web [Hal13a] where there is no central computer *controlling* the Web, no single network on which these protocols work, not even an organization anywhere that *runs* the Web according with Berners-Lee's proposal. For Berners-Lee, the Web is not a material thing, it is completely different than what people could imagine. In [BLF00], he said: "The Web was not a physical *thing* that existed in a certain *place*. It was a *space* in which information could exist" [BLF00]. That is, we are talking about something completely abstract that can be found in everywhere and can be utilized at any time.

There are various technologies that go under the rubric of the Web. In fact, these technologies are related to the infrastructure operation of the Web known as the Internet, *Internationalized Resource Identifier* (IRI), HTTP, HTML, etc. On one hand, the Internet "is a global system of interconnected computer networks that interchange data by packet switching using the standardized Internet Protocol Suite (*Transmission Control Protocol/Internet Protocol* (TCP/IP))" [Sta09] and on the other hand the Web, following the W3C definition, "is an information space in which the items of interest, referred to as resources, are identified by global identifiers called Uniform Resource Identifiers (URI)" [JW04]. In brief, we conclude that the Web acts as an information system based on hypertext pages (developed in the HTML format) to link information through URIs in a global network of computers (Internet) where the hypertext pages, named as Web pages, are requested and transferred over the HTTP [FGM+99] using the hyperlinks that are a reference to a document or specific element within a document.

The linking that we refer to in the last paragraph is one of the main advantage of the Web, because it allows to connect the documents over the Web. This advantage is better

[1]CERN: See `http://www.cern.ch`

highlighted by examining the main principles of the Web architecture.

2.1.1 Principles of Web Architecture

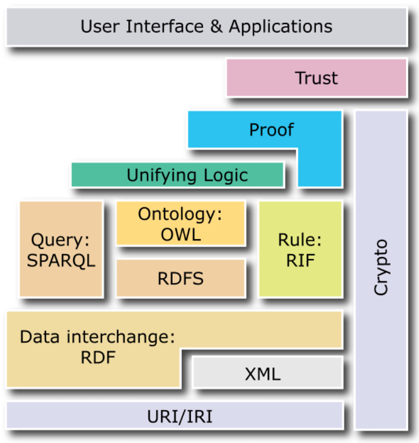
There are five principles that serve as the normative for the Web: Universality, Linking, Self-Description, the Open World, and Least Power [Hal13a]. They are considered as normative but several applications on the Web do not follow them, but it is a recommendation to be under the norm for having a compliance with the Web architecture and the proper operability of the applications.

1. ***Principle of Universality:*** establishes that "*any resource can be identified by a Uniform Resource Identifier (URI)*" (See definitions of resource and URI in Section 2.2). The Universality is based that everything or every concept in the World can be represented and accessible on the Web.
2. ***Principle of Linking:*** establishes that "*any resource can be linked to another resource identified by a URI*". The linking guarantees that all resources are not islands, they have a relationship with other resource.
3. ***Principle of Self-description:*** establishes that "*the obtaining of information for interpreting a Web representation (resource) should be given from the Web representation itself (URI)*". The process of following the links to determine valid interpretations for a resource is informally named *following your nose* in Web architecture [JW04]. This process allows the user-agents to find information that they can use to interpret the Web representation.
4. ***The Open Word Principle:*** establishes that "*the number of resources on the Web can always increase*". Web-pages can appear any moment on the Web, also resources with their respective URIs can be created everytime without any centralized link index.
5. ***Principle of Least Power:*** establishes that "*a Web representation given by a resource should be described in the least powerful but adequate language*". Searching a language that can fulfill the minimal requirements to convey the information and whatever sense and then to extend with more specifications. For example, using HTML, we can build in a simple way the common core of a Web-page and after we can add another technologies to develop more advanced features.

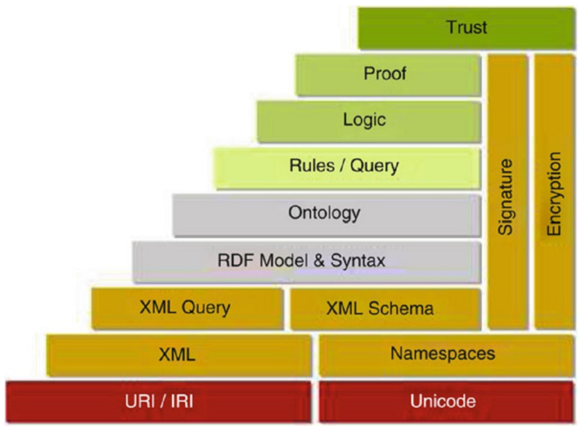
2.1.2 Semantic Web

The Semantic Web (SW) is an extension of the Web beyond the hypertext, the Web evolved from a global information space of linked documents to the SW where documents and data are linked. The SW is also named as the Web of Data because the main goal is to reveal data on the Web in "machine understandable formats" within interlinked datasets. For Berners-Lee et al. in [BLHL+01], "*the Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation*". One of the main standards created for improving the collaboration between computers and people were developed by the W3C, called *eXtensible Markup Language* (XML). XML is used to differentiate content and presentation in Web-pages and helps users to manage, exchange and control the information [BPSM+08]. But also, the SW represents the knowledge for the Web using a special language named RDF (see more details in Section 2.3). We can find several technologies related to the development of the SW, which have been gaining more relevance in the WWW community in the last decade. In [GMB08, TRC+13], the authors show the evolution of the Semantic Web architecture and technologies as URI, XML, RDF, etc.

In Figure 2.1, we present two common versions of the Semantic Web Stack, among others. We choose them because the first one is the Tim Berners-Lee version that shows all the standards for the SW and the second one is considered as the most popular version by Harry Halpin in [Hal13b]. Accordingly with both stacks, we notice that the base of the SW is URI/IRI and the standard model language is RDF to be used for the other technologies. Therefore, we provide more information about them in the following sections.



(a) By Tim Berners-Lee.



(b) By Harry Halpin in [Hal13b].

Figure 2.1: Semantic Web Stack.

2.2 Internationalized Resource Identifier

IRIs are an integral part of the Semantic Web, and constitute the bottom layer supporting the Semantic Web stack, providing the core of linkage and identification capabilities connecting

RDF with resources and the Web. In this section, we introduce the basic concepts of resource, IRI, their types, and related concepts.

2.2.1 Resource

Definition **2.1** . *A resource can be anything that has an identity, [BLFM98, MBLF05, HP09] ranging from a Web-page (with an identifying Web address) to a human being (with an identifying name or social security number). More formally, resources can be organized in two main categories [Hal13b, HP09] (see Figure 2.2): i) information resources (a.k.a. Web IRI, or information IRI) used to designate and access electronic information on the Web (e.g., Web-pages and documents), and ii) non-information resources (a.k.a. Semantic Web IRIs, or non-information IRIs) referring to non-electronic information like physical entities (e.g., person named John, monument named Tour Eiffel, and academic institution named University of Pau) and abstract concepts (e.g., definition of academia, concept of democracy, etc.)*♦

For instance, a Web page describing the University of Pau is an information resource, but the University of Pau itself (i.e., the academic institution) is a non-information resource. Each resource would be identified through its own IRI: using a Web IRI (URL) for the former (e.g., `http://www.univ-pau.fr/live/`) and a Semantic Web IRI for the latter (e.g., `http://dbpedia.org/resource/UPPA`).

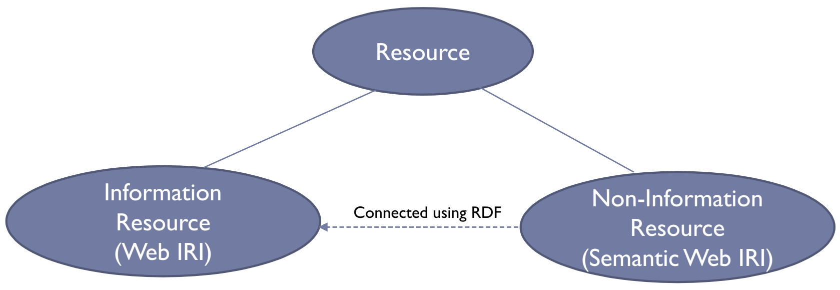

Resource
Information Resource (Web IRI)
Connected using RDF
Non-Information Resource (Semantic Web IRI)

Figure 2.2: Resource Classification

2.2.2 Uniform Resource Identifier - URI

Definition **2.2** . *It is is a string of characters used to identify a resource. A URI can be further classified as a locator (URL), a name (URN), or both [BLFM98, MBLF05] as we show in Figure 2.3*♦

- A *Uniform Resource Locator* (URL) is the most common type of URI, also known as Web address, used to locate information resources on the Web [BLFM98, MBLF05], e.g., `http://dbpedia.org/resource/University`

- A *Uniform Resource Name* (URN) is another form of less commonly used URIs, using the urn scheme of the form *urn:nsi:nss* where *nsi* is the namespace identifier and *nss* a namespace string [BLFM98, MBLF05], e.g., `urn:isbn:0451450523`, providing the ISBN number of a certain book as Romeo et Juliet, but not a reference to the content of the book itself.

2.2.3 Internationalized Resource Identifier - IRI

Definition **2.3** . *It extends the existing Uniform Resource Identifier (URI) scheme to identify a resource, by allowing sequences of characters from the Unicode set, including Chinese, Japanese, and Korean, etc., as opposed to ASCII characters only with URI [DS04]*⧫

There is a mapping from IRIs to URIs, URLs or URNS when IRIs are used instead of them to identify resources. This mapping we show in the Euler Diagram of the relations between IRI, URI, URL, URN in Figure 2.3.

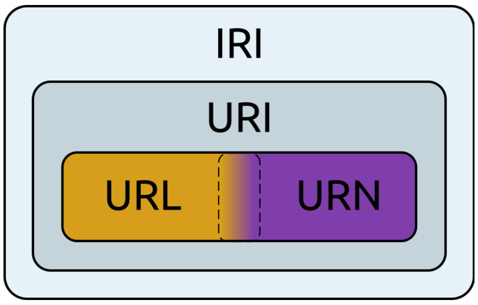


Figure 2.3: This Euler diagram shows that an IRI is a URI, and URI is either a Uniform Resource Locator (URL), a Uniform Resource Name (URN), or both.

2.2.4 IRI Classification

IRIs can be used to designate different things [Boo03], referred to as different IRI types, and which we can classify in two categories: i) conventional IRIs, and ii) owner defined IRIs, as described in Figure 2.4.

1. **Conventional IRI:** An IRI is named based on a standard mechanism for specifying its identity, according to:

   (a) **Implementation:** some Web applications (e.g., DBpedia) give a IRI depending on their implementation (see Figure 2.5.a). This category is classified as following:

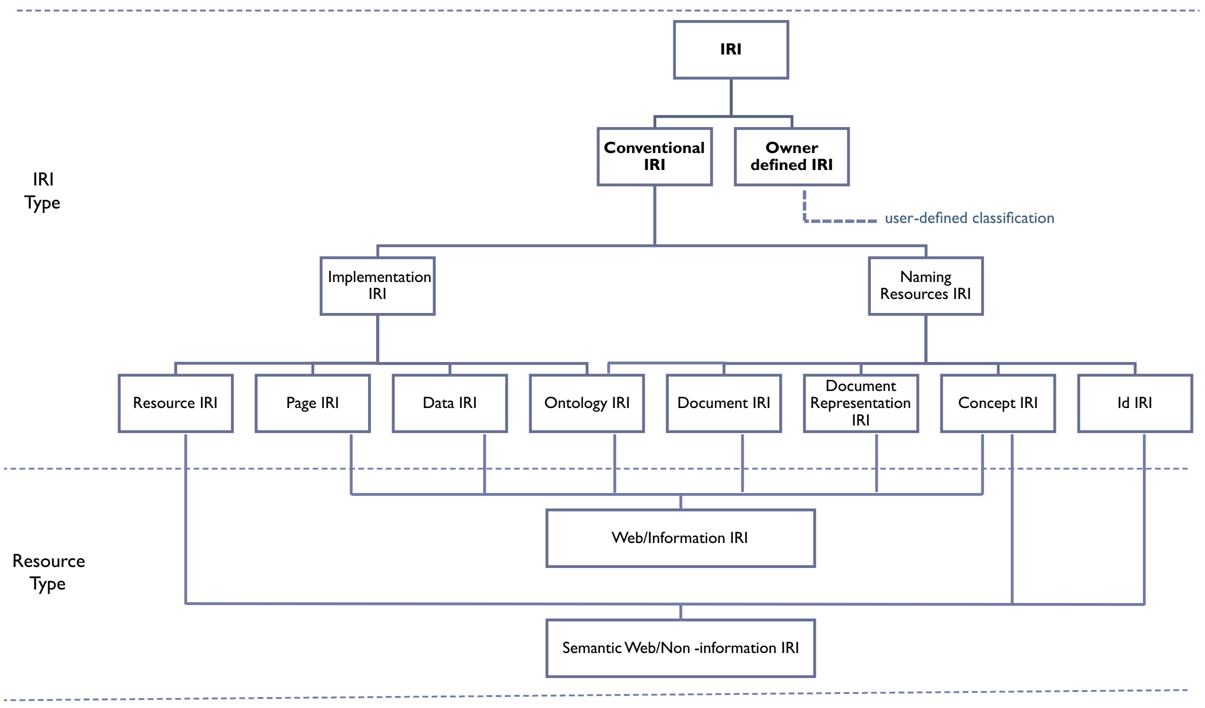


Figure 2.4: Taxonomy of IRI types.

- **Resource**: to name/identify the resource. For example: `http://dbpedia.org/resource/Luxembourg`
- **Page**: to give the Web location of the resource. For example: `http://dbpedia.org/page/Luxembourg`
- **Data**: to provide a representation of the resource (e.g., rdf, nt, json, etc.). For example: `http://dbpedia.org/data/Luxembourg.rdf`
- **Ontology**: to name the domain ontology for the resource. For example: `http://dbpedia.org/ontology/Place`

The IRI structure of each type based on implementation classification is described in Table 2.1.

Table 2.1: Conventional IRI types based on Implementation with their respective structures

IRI type	IRI structure
Resource	`http://{domain}/resource/{resource}`
Page	`http://{domain}/page/{resource}`
Data	`http://{domain}/data/{resource.file-extension}`
Ontology	`http://{domain}/ontologies/{resource}`

(b) **Naming resources:** other Web applications give an IRI using the names of the uses of the IRI (identifier, concept, etc.) [Boo03] as we show in Figure 2.5.b. This category is classified as following:

- **Identifier IRI**: to name/identify the resource. For example: `http://www.example.com/id/sebastien`
- **Document IRI**: to identify the location of the resource. For example: `http://www.example.com/doc/sebastien`
- **Document representation IRI**: each document can have one or more representations (text, html, rdf, owl, etc.). For example: `http://www.example.com/doc/sebastien.rdf`
- **Concept IRI**: to name the concept that models the resource. For example: `http://www.example.com/def/sebastien`
- **Ontology IRI**: to name the domain ontology including the concept that models the resource, e.g., `http://www.example.com/ontologies/person`

The IRI structure of each type based on naming resources classification is described in Table 2.2.

Table 2.2: Conventional IRI types based on naming resources with their respective structures

IRI type	IRI structure
Identifier	`http://{domain}/id/{concept}/{reference}`
Document	`http://{domain}/doc/{concept}/{reference}`
Doc. Representation	`http://{domain}/doc/{concept}/{reference}/{doc.file-extension}`
Concept	`http://{domain}/def/{concept}`
Ontology	`http://{domain}/ontologies/{concept}`

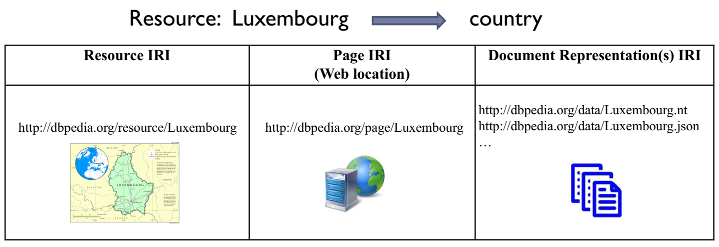
2. **Owner defined IRI:** An IRI is named using a declaration without specifying its type in the path, while using the owner preferences for the naming. For example, `http://toureiffel.fr` is an owner defined IRI since it does not follow any of the conventional IRI types described above, while identifying a certain resource: toureiffel.fr.

2.2.5 IRI Dereferencing

Definition **2.4** . *IRI dereferencing is the process of looking up an IRI on the Web, i.e., accessing the resource referenced by the IRI, which usually comes down to accessing a Web-page or another kind of document representation of the resource available online [BLCC+06, BC06]*♦

Also, note that non-information resources can be linked with information resources available on the Web using RDF (as described in the following sections, cf. Figure 2.2).

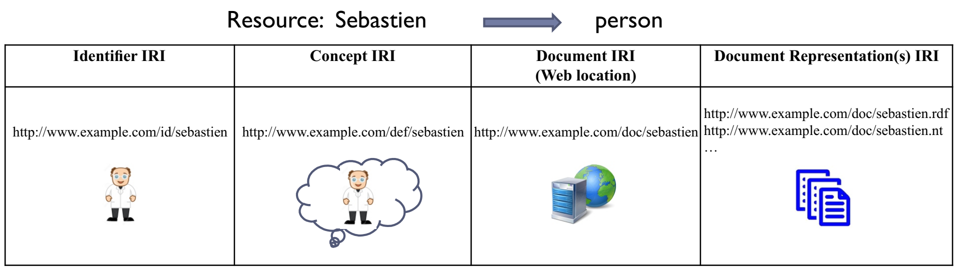
Figure 2.6 shows the dereferencing process of the resource Eiffel Tower in Web-page `http://www.example.org/EiffelTower`. In terms of IRIs, this means issuing an HTTP request in order to retrieve the data pertaining to Eiffel Tower.



Resource: Luxembourg ⟶ country

Resource IRI	Page IRI (Web location)	Document Representation(s) IRI
http://dbpedia.org/resource/Luxembourg	http://dbpedia.org/page/Luxembourg	http://dbpedia.org/data/Luxembourg.nt http://dbpedia.org/data/Luxembourg.json …

(a) *Luxembourg* resource.



Resource: Sebastien ⟶ person

Identifier IRI	Concept IRI	Document IRI (Web location)	Document Representation(s) IRI
http://www.example.com/id/sebastien	http://www.example.com/def/sebastien	http://www.example.com/doc/sebastien	http://www.example.com/doc/sebastien.rdf http://www.example.com/doc/sebastien.nt …

(b) *Sebastien* resource.

Figure 2.5: Examples of IRI uses.

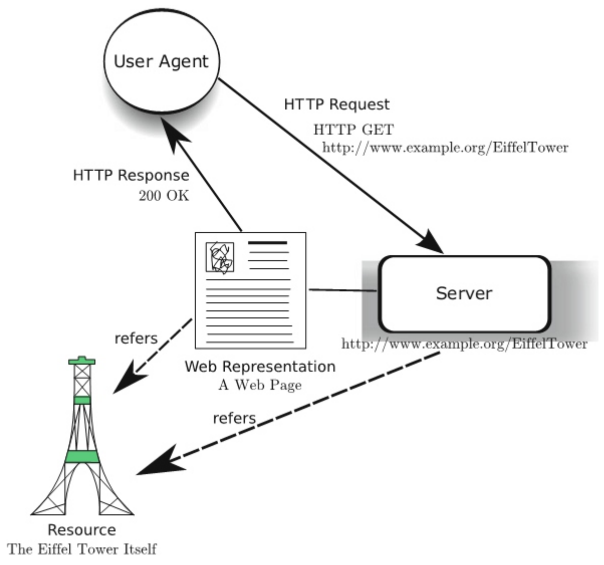


Figure 2.6: Dereferencing the resource Eiffel Tower in [Hal13a].

2.3 Resource Description Framework

The first knowledge representation language for the SW is the Resource Description Framework RDF. RDF was developed by Ora Lassila and Ralph Swick in 1998 [LSWC98] and over the years new versions came out with the intervention of new authors as in [MMM$^+$04, SR14]. The inspiration of this work was based on the *Meta-Content Framework* (MCF) by R.V. Guha [Guh08], who works as a chief of Cyc Project related to the Artificial Intelligence area [GL$^+$92]. RDF was built in accordance with the principles of the Web Architecture explained in Section 2.1.1. To fulfill these principles, RDF (also named as a modeling language) represents the information of resources as assertions in the form *subjet-predicate-object* (named Triples or statements in RDF terminology). To better understand this modeling language, we explain: the RDF terminology, the serialization formats, and the relation with each principle of the Web.

2.3.1 RDF Terminology

DEFINITION **2.5 (RDF Resource** *[r]*) *An RDF resource*[1] *represents the abstraction of an entity (document, abstract concept, person, company, etc.) in the real world. It is noted* $\boldsymbol{r} \in U \cup L$*, where U is a set of IRIs and L is a set of literals. A RDF resource* $\boldsymbol{r}$ *may be associated with a language tag (e.g., @fr, @en, etc.) or with a datatype*[2] *(e.g., string, number, date, etc.) in order to give more information about the corresponding value*⧫

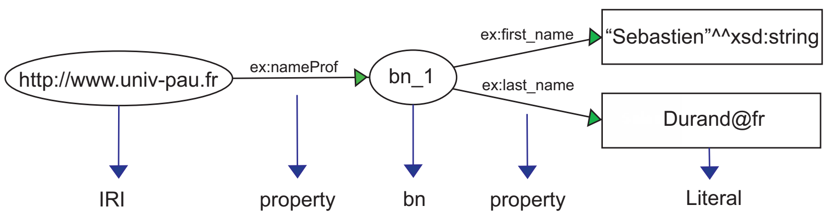


Figure 2.7: Example of RDF graph

Figure 2.7 shows the following RDF resources[3]:

- `http://www.univ-pau.fr` is an IRI that represents the University of Pau;
- "*Sebastien*" ^^*xsd* : *string* is a literal associated with the String datatype; and

[1]The difference between *RDF resource* definition with the definition of *Resource* in 2.1 is that RDF resource is an extension of the resource with more declared characteristics and differenciation between IRIs or literals.

[2]Only literal resources are concerned.

[3]The attributes datatype and language are added to the graph for illustrative and explanatory purposes only.

- *"Durand"@fr* is a literal associated with the French language.

Note that in the remainder of the study, R, Lang and DT are used for naming the set of resources, languages and datatypes respectively.

Definition **2.6 (Blank Node** ***[bn]*****)** *An RDF blank node represents an anonymous RDF resource characterizing a set of RDF resources' properties. A blank node, noted* ***bn*** $\in$ *BN, can be associated with an identifier (or nodeID) to cope with data semantics and simplify the serialization process*[1]♦

For instance, ***bn_1*** in Figure 2.7 is a group of *first name* and *last name* properties. The ***bn*** here is illustrated without an explicit identifier.

Definition **2.7 (Property** ***[p]*****)** *An RDF property is defined as an IRI (conventional or owner defined), noted as* $\mathbf{p} \in U$*, to represent a predicate (relationship) between RDF resources* ***r****, between blank nodes (see Def. 2.6), or both. A data-type and/or a language tag may be declared within a property: utilized to describe the data-type and the language of the associated object literal*♦

Figure 2.7 shows three properties[2] that represent the abstract concepts of a professor's ***full name*** (nameProf), consisting of concepts: ***first name*** and ***last name*** respectively.

Definition **2.8 (Statement** ***[st]*****)** *An RDF statement expresses a relationship between two RDF resources, two blank nodes, or one resource and one blank node. It is defined as an atomic structure consisting of a triple with a Subject (*$\mathbf{s}$*), a Predicate (*$\mathbf{p}$*) and an Object (*$\mathbf{o}$*), noted as* ***st:***$< s, p, o >$*, w.r.t. a specific vocabulary* ***V*** *(see Def. 2.10), where:*

* $\mathbf{s} \in U \cup BN$ *represents the subject to be described,*

* $\mathbf{p} \in U$ *refers to the properties of the subject,*

* $\mathbf{o} \in U \cup BN \cup L$ *describes the object*♦

The example presented in Figure 2.7 underlines 3 statements with different RDF resources, properties, and blank nodes such as:

[1]If no identifier is used, serialization usually provides a meaningless random identifier.

[2]We removed IRIs from *nameProf*, *first_name* and *last_name* to avoid repeating them and thus simplify presentation.

- st_1: <`http://www.univ-pau.fr`, $ex : nameProf$, bn_1 >
- st_2: $<$ bn_1, $ex : first\_name$, *“Sebastien”*^^$xsd : string$ $>$
- st_3: $<$ bn_1, $ex : last\_name$, *“Durand”*$@fr$ $>$

Definition **2.9 (RDF Graph** ***[G]*****)** *An RDF graph is defined as a set of statements denoted by* $G : \{st_1, st_2, st_3, \ldots, st_n\}$ *where* ***G*** *is a directed labeled graph [LS99] in which each statement is represented as a node-edge-node link [KC04]. Therefore,* ***G*** *nodes represent RDF subjects and objects, and linking edges represent corresponding predicates*♦

For instance, Figure 2.7 depicts an RDF Graph made of three statements described following Definition 2.8.

In the remainder of the study, “RDF graph” and “RDF logical representation” are used interchangeably.

Definition **2.10 (RDF Graph Vocabulary** ***[V]*****)** *An RDF Graph Vocabulary is the set of all values occurring in the RDF graph, i.e.,* $V = \boldsymbol{U} \cup \boldsymbol{L} \cup \boldsymbol{BN}$♦

Definition **2.11 (External Vocabulary** ***[QN]*****)** *An RDF External Vocabulary is a set of QNames*[1] *(****QN****) to represent IRI references* $\{qn_1, qn_2, \ldots, qn_n\}$*. Each* $\boldsymbol{qn}_i$ *is a tuple* $< px_i, ns_i >$ *where* $\boldsymbol{px}_i$ *is a prefix*[2] *(e.g., foaf, ex, dc,...) and* $\boldsymbol{ns}_i$ *is a namespace. The prefix is a short name (local*[3] *or standard) that is assigned to a namespace IRI and which can be subsequently referenced in the entire description [MMM+04, SR14]*♦

For instance *QN*={(**ex**, `http://example.org/stuff/1.0`), (**mypx**, `http://ucsp.edu.pe`)}, where “ex” is a standard prefix, “mypx” is a local prefix, and `http://example.org/stuff/1.0` and `http://ucsp.edu.pe/` are the namespaces.

Definition **2.12 (RDF File** ***[F]*****)** *An RDF file is defined as an encoding of a set of RDF statements or of an RDF graph, using a predefined serialization format complying with an RDF W3C standards, such as RDF/XML, Turtle, N3, and others (see Section 2.3.3). Formally:*

$$F = Enc(ST, enc)$$

where:

ST is a set of RDF statements, enc *is the chosen file format following which the statements will be serialized, where enc* $\in$ *{RDF/XML, JSON-LD, Turtle, N-Triple, etc.}*♦

[1] `http://www.w3.org/TR/REC-xml-names/`
[2] Following the W3C Recommendation, we consider that all the prefixes have to be unique for each namespace.
[3] It is a short name of the namespace given by the user.

In the remainder of the study, "RDF file", "RDF serialization" and "RDF physical representation" are used interchangeably.

Table 2.3 summarizes the list of sets used in our approach, based on all these definitions.

Table 2.3: Summarized descriptions of sets used in our approach

Set	Description
U	Set of IRIs
L	Set of literals
BN	Set of blank nodes
ST	Set of Statements
V	Set of IRIs, literals and blank nodes
QN	Set of Qnames
$Lang$	Set of Languages such as en(english), fr(french), es(spanish), etc.
DT	Set of Datatypes such as string, integer, decimal, etc.
NS	Set of Namespaces
Px	Set of Prefixes

2.3.2 RDF and the Principles of the Web

According with the principles of the Web, the RDF standard was developed to fulfill them, as we explain as follows:

1. ***RDF and the Principle of Universality:*** for labelling the nodes and edges RDF uses URIs instead of using natural language terms. So, RDF can model this knowledge making statements that use URIs for identifying the RDF resources. For example, the statement st_1: <`http://www.univ-pau.fr`, $ex:nameProf$, bn_1 > of Figure 2.7 uses the URI `http://www.univ-pau.fr` to identify the *University of Pau*.
2. ***RDF and the Principle of Linking:*** as RDF is composed of RDF resources, and his minimal representation is an RDF statement where two RDF resources are linked by a predicate, then any RDF resource may be linked to another RDF resource. For instance, statement st_1 represents a relation between two resources, where the RDF resource *University of Pau* is linked with the anonymous resource *professor* (represented by a blank node).
3. ***RDF and the Principle of Self-description:*** through the links of an RDF description, we can discover the context of an RDF statement. After discovering the context, we can obtain an interpretation about the RDF resource. Each RDF statement can be transported to several contexts depending on its utilization. So, we can discover the interpretation of the SW data by following the links. For example, in the RDF statement

st_1, one can discover more information about *University of Pau*, like a logo, objectives, name of professors, etc., by accessing `http://www.univ-pau.fr`, we can also have more information about the predicate $ex : nameProf$ following the namespace associated to the RDF resource [Con07].

4. ***RDF and the the Open Word Principle:*** this principle is linked also to the inference on the SW. To help the inference process, a new simple language for declaring sub-classes and sub-properties was developed under the name *RDF Schema* (RDF(S)) based on the RDF standard. In this way, using the RDF statements may infer information, which is a non-trivial problem. Such simple reasoning uses a set of axiomatic RDF statements, rules for inferences, and semantic conditions to infer more RDF statements [Hay04, HPS14]. For handling complex inferences, *Web Ontology Language* (OWL) [PSHH$^+$04] also appears as an extension of RDF semantics, that allows to handle restrictions with cardinality in predicates, subjunctions, disjunctions, etc.

5. ***RDF and the Principle of Least Power:*** since RDF is a language designed to build the SW using the languages of triples (RDF statements) as the most basic language, hence we conclude that RDF can be considered as the least powerful and simple language to develop the SW.

2.3.3 Serialization Formats

Over the Web, there are several serialization formats for RDF descriptions. The most popular and utilized is the RDF/XML format [Bec04] because it is based on the XML standard which is a fundamental standard for efficient data management and exchange over the Web [BPSM$^+$08]. For this reason, we classify these formats in two categories: i) XML serialization formats and ii) Non-XML serialization formats. The differences between them lie in verbosity, compression, and human-understandability, among other aspects.

- ***XML serialization formats:*** There is the typical RDF/XML format (e.g., in Figure 2.8), an abbreviated form of RDF/XML (e.g., in Figure 2.9) format, and a simplified format designed to be integrated with Web page formatting (e.g., HTML) calledRDFa [ABMP08] (e.g, in Figure 2.10).

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:ex="http://example.org/stuff/1.0/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <rdf:Description rdf:nodeID="bn_1">
  <ex:first_name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Sebastien</ex:first_name>
  <ex:last_name xml:lang="fr">Durand</ex:last_name>
 </rdf:Description>
 <rdf:Description rdf:about="http://www.univ-pau.fr">
  <ex:nameprof rdf:nodeID="bn_1"/>
 </rdf:Description>
</rdf:RDF>
```

Figure 2.8: RDF/XML serialization of the RDF graph depicted in Figure 2.7

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:ex="http://example.org/stuff/1.0/">
 <rdf:Description rdf:about="http://www.univ-pau.fr">
  <ex:nameprof>
   <rdf:Description rdf:nodeID="bn_1">
    <ex:first_name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Sebastien</ex:first_name>
    <ex:last_name xml:lang="fr">Durand</ex:last_name>
   </rdf:Description>
  </ex:nameprof>
 </rdf:Description>
</rdf:RDF>
```

Figure 2.9: Abbreviated RDF/XML serialization of the RDF graph depicted in Figure 2.7

```
<div xmlns="http://www.w3.org/1999/xhtml"
 prefix="
  rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
  ex: http://example.org/stuff/1.0/
  xsd: http://www.w3.org/2001/XMLSchema#
  rdfs: http://www.w3.org/2000/01/rdf-schema#"
 >
 <div typeof="rdfs:Resource" about="http://www.univ-pau.fr">
  <div rel="ex:nameprof">
   <div typeof="rdfs:Resource">
    <div property="ex:last_name" xml:lang="fr" content="Durand"></div>
    <div property="ex:first_name" datatype="xsd:string" content="Sebastien"></div>
   </div>
  </div>
 </div>
</div>
```

Figure 2.10: RDFa serialization of the RDF graph depicted in Figure 2.7

- ***Non-XML serialization formats:*** There are several formats such as *Notation 3* (N3) [BL05] (e.g., in Figure 2.13), JSON-LD[1] [SLK$^+$14](e.g, in Figure 2.12), Turtle [BBL11] (e.g., in Figure 2.14), and N-Triples [BB01] (e.g., in Figure 2.11).

[1]Note that JSON-LD has different serializations: compacted, flattened, expanded and embedding in HTML.

```
{
 "@context": {
   "ex": "http://example.org/stuff/1.0/",
   "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
   "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
   "xsd": "http://www.w3.org/2001/XMLSchema#"
 },
 "@graph": [
   {
    "@id": "http://www.univ-pau.fr",
    "ex:nameprof": {
      "@id": "_:N398bc0d5afc1498087d03f880cf3a577"
    }
   },
   {
    "@id": "_:N398bc0d5afc1498087d03f880cf3a577",
    "ex:first_name": "Sebastien",
    "ex:last_name": {
      "@language": "fr",
      "@value": "Durand"
    }
   }
 ]
}
```

Figure 2.12: JSON-LD serialization of the RDF graph depicted in Figure 2.7

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/stuff/1.0/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://www.univ-pau.fr> ex:nameprof [ ex:first_name "Sebastien"^^xsd:string ;
        ex:last_name "Durand"@fr ] .
```

Figure 2.13: N3 serialization of the RDF graph depicted in Figure 2.7

```
<http://www.univ-pau.fr> <http://example.org/stuff/1.0/nameprof> _:genid1 .
_:genid1 <http://example.org/stuff/1.0/first_name>
"Sebastien"^^<http://www.w3.org/2001/XMLSchema#string> .
_:genid1 <http://example.org/stuff/1.0/last_name> "Durand"@fr .
```

Figure 2.11: N-Triple serialization of the RDF graph depicted in Figure 2.7

For ease of presentation, we adopt the RDF/XML format to illustrate RDF serialization results in the remainder of our research, given that RDF/XML: i) has been promoted on the Web as the W3C standard format for RDF, ii) is more flexible[1] and structured than other formats, and thus iii) can be easily used for conversion between formats like Turtle, N3, etc.

[1]Referring to many ways for specifying the statements.

```
@prefix ns0: <http://example.org/stuff/1.0/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://www.univ-pau.fr> ns0:nameprof [
  ns0:first_name "Sebastien"^^xsd:string ;
  ns0:last_name "Durand"@fr
 ] .
```

Figure 2.14: Turtle serialization of the RDF graph depicted in Figure 2.7

2.4 The Web of Linked Data

First of all, we have to highlight that the Web of Linked Data is a constituent part of the SW. Linked Data appears as a solution due to the increase of information on the Web, where the goal is not only connecting HTML documents (URLs), but also connecting data (information inside of these documents) [HB11], based on three technologies: URI, RDF and HTTP. So, we can consider that the Linked Data paradigm is a guideline highlighting the best practices for publishing and connecting structure Data on the Web (using links between data from different sources) [BHBL09]. It promotes the development and support of a self-sustaining ecosystem related to the publication and usage of data on the Web, where data should be easily discoverable and understandable by humans and machines alike, facilitating data interaction between publishers and consumers [FLBC16]. Hence, the Web of Linked Data adds an extra value to the traditional Web because the concept goes beyond linking only documents, toward linking resources [BLCC$^+$06].

To disseminate the goal of Linked Data, the W3C Semantic Web Education and Outreach Group supports the creation of the community project LOD[1] [BHIBL08], that was founded in 2007. This community aims to bootstrap the SW data with Linked Data by identifying existing datasets that are available, converting these datasets to RDF according to their principles, and publishing them on the Web [BHBL09]. Inside of this community, they develop several projects as DBpedia[2] (based on extract information of Wikipedia and make this information available on the Web), LinkedGeoData[3] (based on extract information of the OpenStreetMap project and make this information available on the Web), and FOAF[4] (that is a dictionary of people-related terms that can be used in structured data on the Web). Also, thanks to the LOD project, various interesting open datasets as DBLP[5], Geo-names[6] and WorNet[7] available on the Web [BHAR07]. We show the evolution of these projects and the relationship between them and other datasets in Section 2.4.2.

[1] http://linkeddata.org
[2] http://dbpedia.org/about
[3] http://linkedgeodata.org/About
[4] http://www.foaf-project.org/
[5] http://dblp.uni-trier.de/db/
[6] http://www.geonames.org/
[7] http://wordnet.princeton.edu/online/

2.4.1 Principles

For Berners-Lee in [BL06], the Web of Data (SW) has four principles to publish the data. Following these principles the data became in one big data space, where all the information is linked.

1. Use URIs as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL[1] [HSP13])
4. Include links to other URIs, so that people can discover related things.

The principles give us the guidelines for publishing and connecting data following the standards and using the infrastructure of the Web.

2.4.2 Linked Open Data Cloud Evolution

The evolution of the LOD is exponential since its inception to date. For example:

- As of May 2009, the LOD cloud consists of over 4.7 billion RDF triples, which are interlinked by around 142 million RDF links [LAH$^+$09].
- As of November 2010, the LOD cloud consists of over 26.93 billion RDF triples, which are interlinked by around 395 million RDF links (see Table 2.4).
- As of September 2011, the LOD cloud consists of over 31.643 billion RDF triples, which are interlinked by around 504 million RDF links (see Table 2.5).

Therefore, starting by 12 smaller datasets in 2007 until having 570 datasets in 2014, the LOD cloud has had an important growing until nowadays, not only with respect to the quantity of datasets as we show in Figures 2.15 and 2.16, but also with respect to the quantity of RDF triples and RDF links related to some areas as Publications, Life sciences, etc (e.g., see Tables 2.4 and 2.5). The LOD project has a huge importance for Web applications users and a big impact on the Web community.

For instance, in Figure 2.17, we show a wikipedia Web-page of Luxembourg and its corresponding DBpedia Web-page. The Dbpedia Web-page refers to several statements that

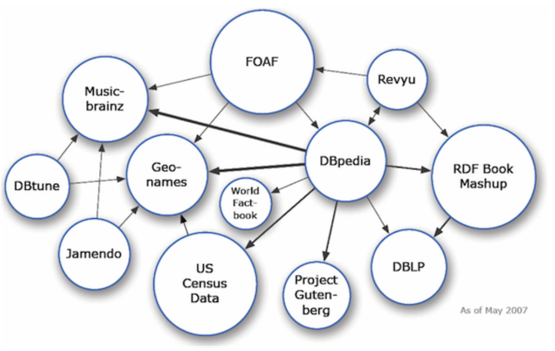
[1]SPARQL is the standard query language for RDF.

Table 2.4: Topology of the Web of Data - November 2010 (Source: http://lod-cloud.net/, 2010)

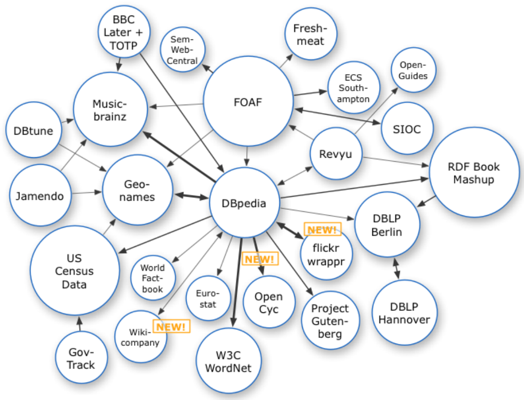
Domain	Data Sets	Triples	Percent	RDF Links	Percent
Cross-domain	20	1,999,085,950	7.42	29,105,638	7.36
Geographic	16	5,904,980,833	21.93	16,589,086	4.19
Government	25	11,613,525,437	43.12	17,658,869	4.46
Media	26	2,453,898,811	9.11	50,374,304	12.74
Libraries	67	2,237,435,732	8.31	77,951,898	19.71
Life sciences	42	2,664,119,184	9.89	200,417,873	50.67
User Content	7	57,463,756	0.21	3,402,228	0.86
	203	26,930,509,703		395,499,896	

Table 2.5: Topology of the Web of Data - September 2011 (Source: http://lod-cloud.net/, 2011)

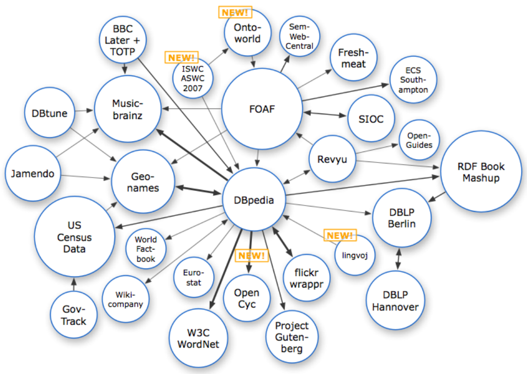
Domain	Data Sets	Triples	Percent	RDF Links	Percent
Cross-domain	41	4,184,635,715	13.23	63,183,065	12.54
Geographic	31	6,145,532,484	19.43	35,812,328	7.11
Government	49	13,315,009,400	42.09	19,343,519	3.84
Media	25	1,841,852,061	5.82	50,440,705	10.01
Publications	87	2,950,720,693	9.33	139,925,218	27.76
Life sciences	41	3,036,336,004	9.60	191,844,090	38.06
User Content	20	134,127,413	0.42	3,449,143	0.68
	295	31,634,213,770		503,998,829	



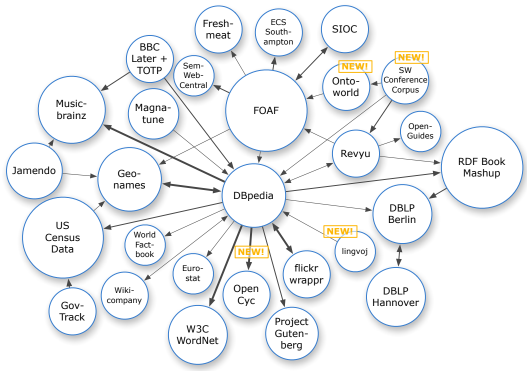
(a) 01 of May



(b) 08 of October



(c) 07 of November



(d) 10 of November

Figure 2.15: Linked Datasets as 2007 (Source: http://lod-cloud.net/, 2007).

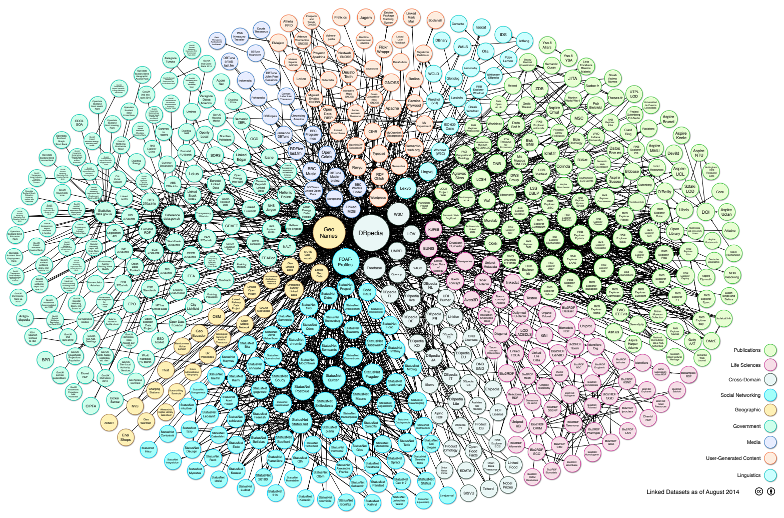


Figure 2.16: Linked Datasets as 30 of August 2014 (Source: http://lod-cloud.net/, 2014).

are linked to different vocabularies where concepts allow for example: to do inferences, create more information, etc. Therefore, it proves that connecting structure data provides the users a powerful tool for publishing and sharing their information on the LOD cloud in several contexts. Through this interconnection of data, people can obtain new knowledge, more details and different perspectives about a resource.

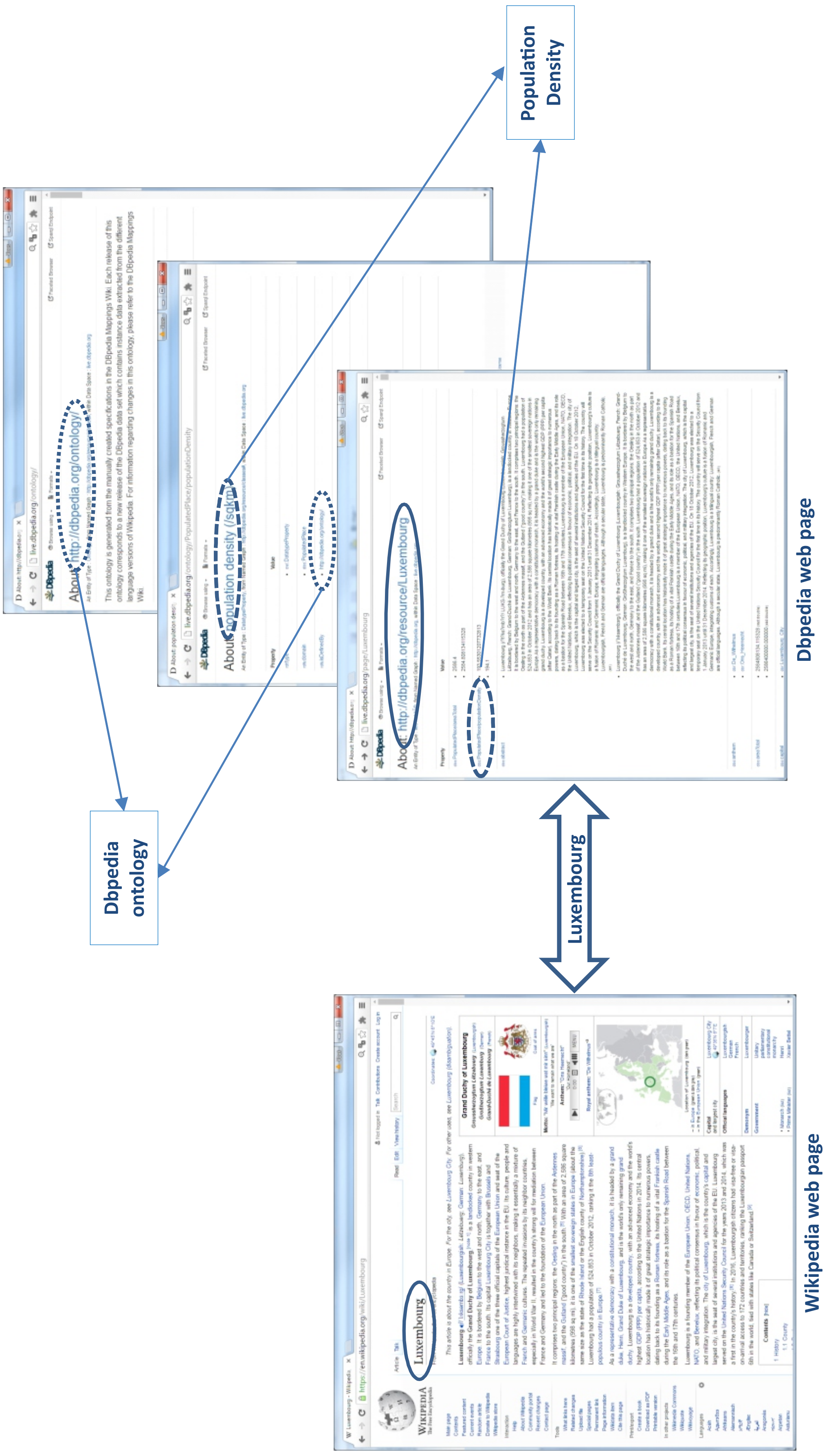

Population Density
Dbpedia ontology
About: http://dbpedia.org/ontology/
About: population density (/sqkm)
About: http://dbpedia.org/resource/Luxembourg
Luxembourg
Dbpedia web page
Wikipedia web page

Figure 2.17: Example of Luxembourg in LOD cloud.

2.5 Summary

In this chapter we have introduced all the background necessary for understanding the concepts and Web technologies linked to RDF techonology and the Semantic Web.

We began this chapter with a brief introduction about World Wide Web and its relation with the Semantic Web (Section 2.1). We then described the two main technologies for the Semantic Web: IRIs (Section 2.2) and RDF (Section 2.3). Finally, in Section 2.4, we discussed the Linked data movement which has given rise to the Web of Linked Data and boosted the increase of RDF triples and their use on the Web.

Against this background, in the next chapter, we identify a number of research challenges related to normalization through small use cases (synthetics and real).

Chapter 3

Motivating Examples

"Those who know, do. Those who understand, teach."

— Aristotle

In this chapter, we present short scenarios describing RDF descriptions related to University of Pau and Luxembourg country. Each scenario aims to illustrate different problems related to duplicated or non-used information. For a better explanation, we categorize the motivations of our study into four different levels:

- **Logical redundancies**, where multiple RDF statements (i.e., triples), including redundant subjects, predicates, and/or objects, describe the same information (Section 3.1),
- **Physical disparities**, where different serializations, including duplicated namespaces and distinct literal representations, describe the same initial RDF graph (Section 3.2),
- **Semantic ambiguities**, where multiple RDF statements, including redundant blank nodes (as subjects or objects) and/or literals (as objects), describe the same semantic information (Section 3.3). Note that semantic information refers not only to the value of the node itself, but rather to the meaning of the whole statement: such that the meaning of a literal and/or blank node (likewise for IRI nodes/edges) depends on the subject and/or predicate of the containing statement,
- **IRI discrepancies**, where different IRIs, including duplicated IRIs with different types and coreferences, describe the same resource (entity) (Section 3.4).

In all these sections, we present different RDF statements which can represent the same information. This inherently renders such RDF files difficult to process via automated software applications and humans alike.

We follow each level with a small explanation of the use case and the challenges present for RDF Normalization.

3.1 Logical (Graph) Redundancies

3.1.1 Use Case 1 - University of Pau (Logical Representation)

For this use case (synthetic), we use three RDF resources:

- University of Pau `http://www.univ-pau.fr` (IRI),
- Professor (Sebastien Durand), and
- Laboratory UPPA `http://liuppa.univ-pau.fr/live` (IRI).

We represent these resources across a set of triples in Figure 3.1. The RDF resource *professor* in the RDF graph in Figure 3.1 shows two blank nodes *bn*_1 and UX having a name consisting of a first name "Sebastien" (literal) associated with the string datatype and a last name "Durand" (literal) associated with a French language tag.

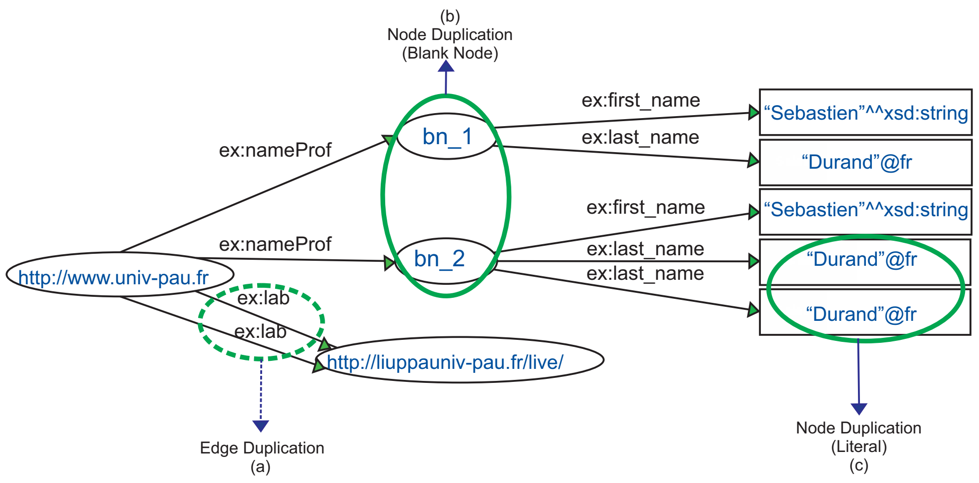


Figure 3.1: RDF Graph 1 with node and edge duplication.

3.1.2 Challenges in Use Case 1

Considering use case 1, we show different kinds of redundancies in the RDF logical representation as follows:

- **Problem 1 - Edge Duplication**: where identical edges, designating identical RDF predicates, appear more than once (Figure 3.1.a).

- **Problem 2 - Node duplication**: where identical nodes, designating identical RDF subjects and/or objects, appear more than once. For instance, Figure 3.1.b highlights *blank node* duplication with two kinds of representations: one with a nodeId identifier called "UX" and the other without nodeId called "bn_1" (describing the same node in the RDF graph), whereas Figure 3.1.c highlights *literal* duplications.

  Note that certain features of RDF literals, represented as nodes in the RDF Graph 1, may also result in special node duplications as in Figure 3.2 which is a variation of the RDF Graph 1:

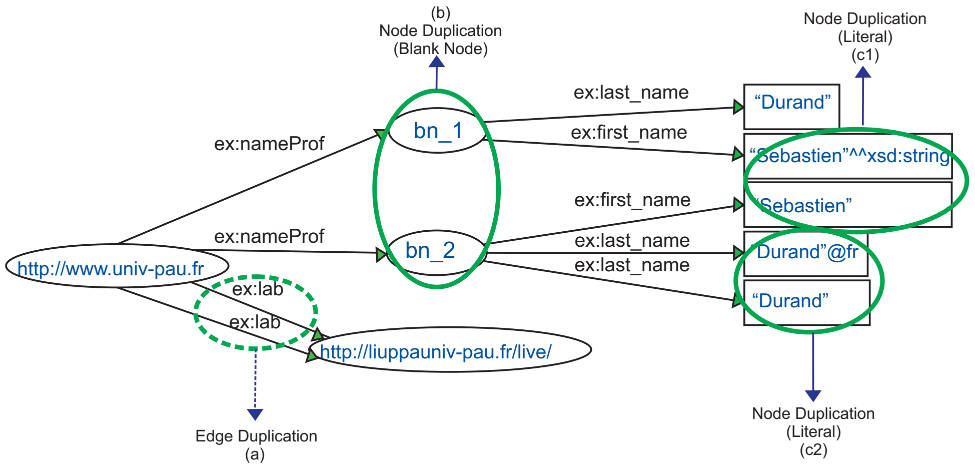


Figure 3.2: RDF Graph 2 with node and edge duplication with and without datatypes and languages.

  - **Problem 2.1 - Handling data-typed Node literals**: Literals can be typed or not (e.g., Figure 3.2.c1 represents the same element with and without datatype definitions).

  - **Problem 2.2 - Handling language-tagged Node literals**: Distinguishing between identical literals having different language tags, when language tags are available (e.g., Figure 3.2.c2 represents the same element with and without language tags).

- **Problem 3 - Edge and node duplication**: where both edge and node duplication problems affect the same statements.

```
1 <?xml version="1.0" encoding="UTF-8" encoding="UTF-8" standalone="no"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3   xmlns:dc="http://purl.org/dc/elements/1.1/"
4   xmlns:ex="http://example.org/stuff/1.0/"
5   xmlns:ex1="http://example.org/stuff/1.0/"
6   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
7 <rdf:Description rdf:about="http://www.univ-pau.fr">
8    <ex:nameProf rdf:nodeID="UX"/>
9    <ex:nameProf>
10    <rdf:Description>
11      <ex1:first_name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Sebastien</ex:first_name>
12      <ex:last_name xml:lang="fr">Durand</ex:last_name>
13      <ex:last_name xml:lang="fr">Durand</ex:last_name>
14     </rdf:Description>
15    </ex:nameProf>
16    <ex:lab rdf:resource="http://liuppa.univ-pau.fr/live/"/>
17    <ex:lab rdf:resource="http://liuppa.univ-pau.fr/live/"/>
18  </rdf:Description>
19  <rdf:Description rdf:nodeID="UX">
20    <ex1:first_name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Sebastien</ex:first_name>
21    <ex:last_name xml:lang="fr">Durand</ex:last_name>
22  </rdf:Description>
23 </rdf:RDF>
```

Figure 3.3: RDF/XML serialization of the RDF graph in Fig. 3.1.

3.2 Physical (Serialization) Disparities

3.2.1 Use Case 2 - University of Pau (Physical Representation)

Use case 2 represents a possible serialization of the RDF Graph 1 developed in use case 1 (Section 3.1.1) in Figure 3.1. RDF Graph 1 is encoded in RDF/XML format in Figure 3.3, to show the namespaces linked with the resources, the order of the statements, and the type of format in Figure 3.3.

3.2.2 Challenges in Use Case 2

Considering Figure 3.3, one can see that several types of redundancies and disparities are introduced: some are inherited from the logical level (node duplication in lines 8 and 9 and edge duplication in lines 16 and 17), while others appear at the physical (serialization) level, namely:

```
1 <rdf:RDF
2  xmlns:dc="http://purl.org/dc/elements/1.1/"
3   xmlns:ex="http://example.org/stuff/1.0/"
4   xmlns:ex1="http://example.org/stuff/1.0/"
5   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
6 <rdf:Description rdf:about="http://www.univ-pau.fr">
7    <ex1:nameProf rdf:parseType="Resource">
8      <ex:last_name>Durand</ex:last_name>
9      <ex:first_name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Sebastien</ex:first_name>
10    </ex:nameProf>
11    <ex:nameProf rdf:parseType="Resource">
12      <ex:first_name>Sebastien</ex:first_name>
13      <ex:last_name xml:lang="fr">Durand</ex:last_name>
14      <ex:last_name>Durand</ex:last_name>
15    </ex:nameProf>
16    <ex:lab rdf:resource="http://liuppa.univ-pau.fr/live/"/>
17    <ex:lab rdf:resource="http://liuppa.univ-pau.fr/live/"/>
18   </rdf:Description>
19 </rdf:RDF>
```

Figure 3.4: RDF/XML serialization of the RDF graph in Fig. 3.2.

- **Problem 4 - Namespace duplication**: where two different prefixes are used to designate the same namespaces, e.g., *ex* and *ex1* addressing the namespace of `http://example.org/stuff/1.0/` (lines 4-5 in Figure 3.3),

- **Problem 5 - Unused namespace**: where one (or more) namespace(s) is (are) declared but never called in the body of the document, e.g., *dc* (line 3 in Figure 3.3).

Note that other kinds of features may also result in RDF serialization disparities as illustrated in Figure 3.4, showing namely:

- **Problem 6 - Handling node order variation**: i.e., node siblings in the RDF description might be ordered differently when serialized (e.g., nodes in lines 6-18 in Figure 3.4 follow the order of appearance of XML elements different to the order in lines 7-22 in Figure 3.3, which can be re-ordered differently in another serialization).

- **Problem 7 - Handling serialization format variation**: i.e., RDF elements in the same RDF description might be formatted differently when serialized (e.g., blank nodes in lines 8,19-22 in Figure 3.3 follow the flat RDF/XML serialization, compared with nodes in lines 7-10 in Figure 3.4 that follow the compact RDF/XML serialization)

3.3 Semantic Ambiguities

3.3.1 Use Case 3 - University of Pau (Logical Representation)

This use case is a variation of RDF Graph 1 developed in use case 1 (Section 3.1.1) in Figure 3.1. RDF Graph 3 (Figure 3.5) shows the following additional information about:

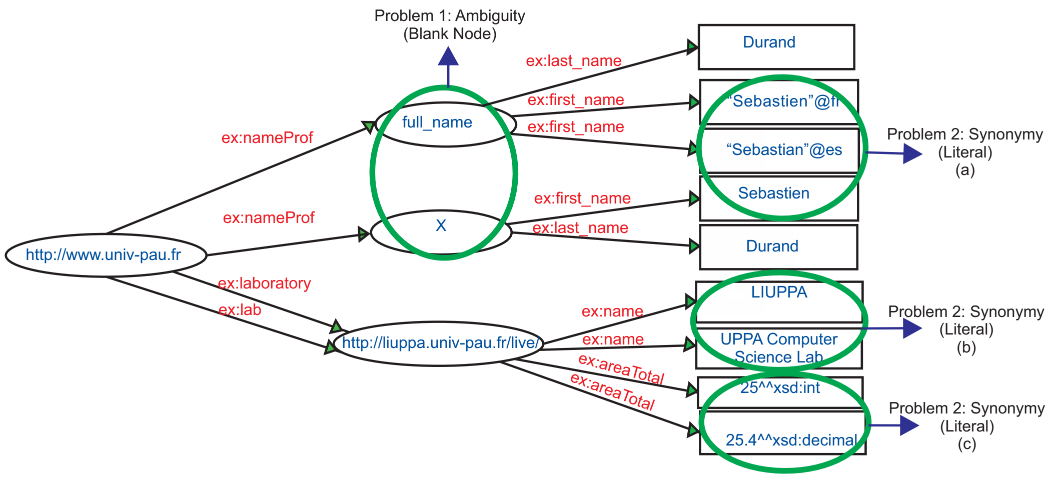


Figure 3.5: RDF Graph 3 with semantic ambiguities

- Different languages tags for the literal "Sebastien" (french and spanish),
- One Predicate between `http://www.univ-pau.fr` and `http://liuppa.univ-pau.fr/live/` called *lab*,
- The name of the laboratory, expressed in two different ways: *LIUPPA* and *UPPA Computer Science Lab*, and
- The area Total of the laboratory, expressed with two different datatypes and formats: 25 as integer and 25.4 as decimal.

3.3.2 Challenges in Use Case 3

Here, one can easily see several kinds of semantic ambiguities:

- **Problem 8 - Ambiguity in Blank Nodes**: where different blank nodes (with or without nodeIds), designating RDF subjects and/or objects, describe the same information (Figure 3.5),

- **Problem 9 - Synonymy in Literals**: where different literals, designating RDF objects, describe the same information (i.e., using acronyms, different languages, different datatypes, etc.), e.g., Figure 3.5.b highlights literal synonymity using acronyms.

  Here, note that certain properties of RDF literals, namely language tag and/or datatypes, may also result in special literals synonymy:

  – **Problem 9.1 - Handling language-tagged literals**: Literals can be assigned a language tag or not, e.g. Figure 3.5.a represents the same element but in different languages while specifying the language tag in the first two cases while omitting it in the third case.

  – **Problem 9.2 - Handling data-typed literals**: Literals can also be assigned different datatypes, e.g. Figure 3.5.c represents the same element but using the *int* type or the *decimal* type, following the number format.

Note that we consider literal/blank node duplications within their contexts in the corresponding RDF statements. For instance, deciding on whether or not the value of a given literal node (e.g., "LIUPPA") within the RDF statement consists of a duplication of another (e.g., "UPPA Computer Science Lab"), depends on the statement as a whole (e.g., $<$ `http://liuppa.univ-pau.fr/live/`, ex : $name$, $LIUPPA$ $>$ and $<$ `http://liuppa.univ-pau.fr/live/`, ex : $name$, *UPPA Computer Science Lab* $>$ in Figure 3.5). This is different from the situation where the authors of the RDF statements would like to emphasize the fact that the mentioned lab has two synonymous names. In the latter situation, they would formulate the statements differently, e.g., $<$ `http://liuppa.univ-pau.fr/live/`, ex : $name$, $LIUPPA$ $>$ and $<$ `http://liuppa.univ-pau.fr/live/`, ex : $altName$, *UPPA Computer Science Lab* $>$, which would not be considered as duplicates (following our approach) and would be preserved in the normalized RDF output. Likewise for the other cases mentioned in Problems 8 to 15, where we target unintentional duplications (which ought to be eliminated) and not user intended ones (which will be preserved).

However, to avoid any confusions (i.e., deleting duplicates otherwise deemed relevant by the authors/users), an interactive verification phase can be added in our normalization process, prompting the user whenever the system detects duplication cases covering Problems 8 to 15, so that the system proceeds according to the user's input (i.e., unintentional duplication: to be removed, or intentional duplication: to be preserved).

3.3.3 Semantic Ambiguities creating Logical (Graph) Redundancies

Consider the example given in Figure 3.6 in which we describe the University of Pau `http://www.univ-pau.fr` (IRI) having a professor (bn) with first name *Sebastien* (literal) and last

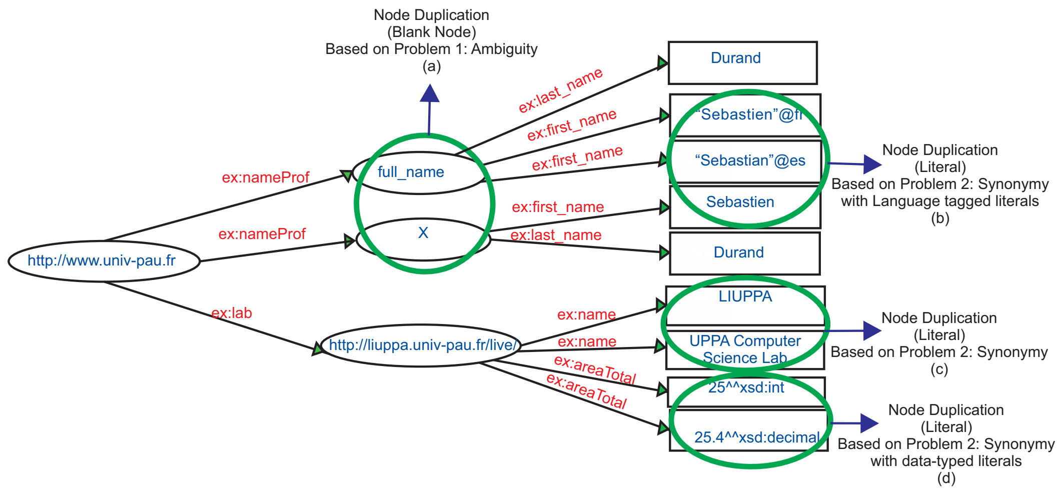


Figure 3.6: RDF Graph 4 based on RDF Graph 3 with Logical redundancies due to semantic ambiguities (concerning problems 1 and 2)

name *Durand* (literal), and a *laboratory* (IRI) with name *LIUPPA* (literal) and total area *25* (literal) highlighting different logical redundancies in the form of RDF graph node duplications:

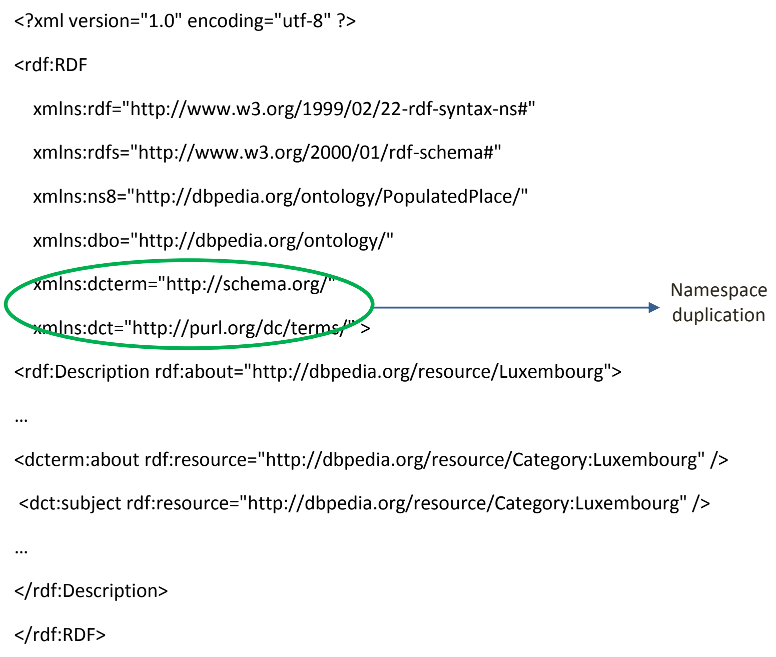
- **Problem 10 - Node Duplication based on Semantic Ambiguities:** where semantically equivalent nodes, designating semantically equivalent subjects and/or objects, appear more than once, e.g., Figure 3.6.a highlights a blank node duplication and Figure 3.6.b, c and d show different kinds of literal node duplications.

3.4 IRI Discrepancies

3.4.1 Use Case 4 - Luxembourg Country (Logical Representation)

This use case is based on part of a Dbpedia real RDF graph. The RDF Graph 5 represents the RDF resource: Luxembourg with different types of IRIs and literals and two ontology[1] descriptions for its description in Figure 3.7.

[1]`http://dbpedia.org/ontology/Place` and `http://dbpedia.org/ontology/Location` are two descriptions detailed on DBpedia ontology.



```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:ns8="http://dbpedia.org/ontology/PopulatedPlace/"
   xmlns:dbo="http://dbpedia.org/ontology/"
   xmlns:dcterm="http://schema.org/"
   xmlns:dct="http://purl.org/dc/terms/" >
<rdf:Description rdf:about="http://dbpedia.org/resource/Luxembourg">
…
<dcterm:about rdf:resource="http://dbpedia.org/resource/Category:Luxembourg" />
 <dct:subject rdf:resource="http://dbpedia.org/resource/Category:Luxembourg" />
…
</rdf:Description>
</rdf:RDF>
```

Namespace duplication

Figure 3.8: Sub-part of RDF Serialization for the RDF Graph 5 in Figure 3.7 with Physical disparities due to IRI discrepancies (concerning problems 11 and 12).

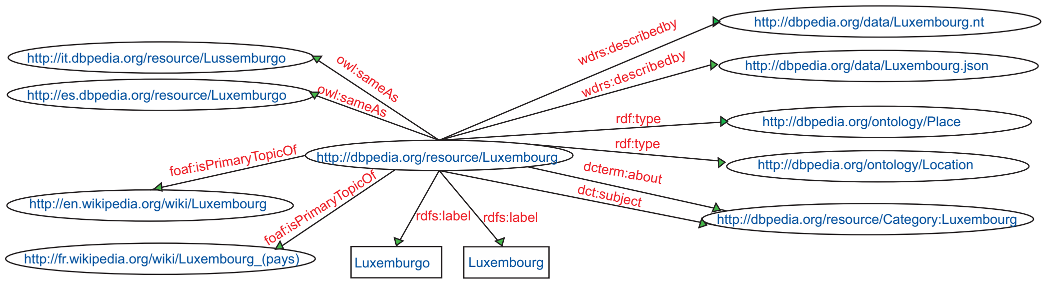

http://dbpedia.org/resource/Luxembourg
http://it.dbpedia.org/resource/Lussemburgo
http://es.dbpedia.org/resource/Luxemburgo
owl:sameAs
owl:sameAs
foaf:isPrimaryTopicOf
foaf:isPrimaryTopicOf
http://en.wikipedia.org/wiki/Luxembourg
http://fr.wikipedia.org/wiki/Luxembourg_(pays)
rdfs:label
rdfs:label
Luxemburgo
Luxembourg
wdrs:describedby
wdrs:describedby
http://dbpedia.org/data/Luxembourg.nt
http://dbpedia.org/data/Luxembourg.json
rdf:type
rdf:type
http://dbpedia.org/ontology/Place
http://dbpedia.org/ontology/Location
dcterm:about
dct:subject
http://dbpedia.org/resource/Category:Luxembourg

Figure 3.7: RDF Graph 5 about of Luxembourg RDF resource

3.4.2 Use Case 5 - Luxembourg Country (Physical Representation)

This use case 5 represents a possible serialization of the RDF Graph 5 developed in the use case 4 (Section 3.4.1) in Figure 3.7. The RDF Graph 5 is encoded in RDF/XML format to show the namespaces linked with the resources in Figure 3.8.

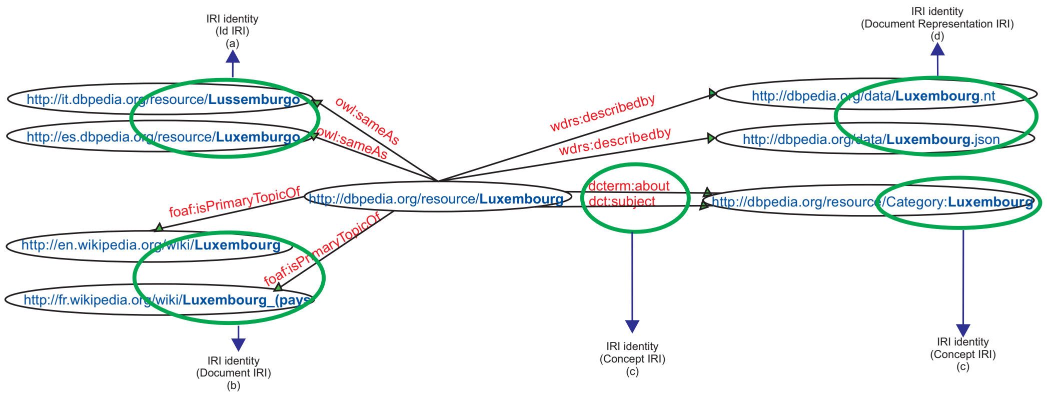

IRI identity
(Id IRI)
(a)
http://it.dbpedia.org/resource/Lussemburgo
http://es.dbpedia.org/resource/Luxemburgo
owl:sameAs
owl:sameAs
foaf:isPrimaryTopicOf
http://dbpedia.org/resource/Luxembourg
foaf:isPrimaryTopicOf
http://en.wikipedia.org/wiki/Luxembourg
http://fr.wikipedia.org/wiki/Luxembourg_(pays
IRI identity
(Document IRI)
(b)
wdrs:describedby
wdrs:describedby
dcterm:about
dct:subject
IRI identity
(Concept IRI)
(c)
IRI identity
(Document Representation IRI)
(d)
http://dbpedia.org/data/Luxembourg.nt
http://dbpedia.org/data/Luxembourg.json
http://dbpedia.org/resource/Category:Luxembourg
IRI identity
(Concept IRI)
(c)

Figure 3.9: RDF Graph with IRI discrepancies - IRI identity

3.4.3 Challenges in Use Cases 4 and 5

Consider Figures 3.9 and 3.10 which represent the RDF graph 5 with severals IRIs describing the same resource (e.g., Luxembourg), such that Fig. 3.9 highlights an IRI identity problem, whereas Figure 3.10 reflects an IRI coreference problem. In other words, several types of identities (in Figure 3.9) and references (in Figure 3.10) are introduced to give extra information about one resource, but not all the IRIs have the same information of the resource.

- **Problem 11 - IRI Identity**: where two different IRIs are used to designate in a different way the same resource. Consider for instance the case of DBpedia describing the resource "Luxembourg" in Figure 3.9. For example, `http://dbpedia.org/resource/Luxembourg`, `http://en.wikipedia.org/wiki/Luxembourg`, and `http://dbpedia.org/data/Luxembourg.nt` (cf. Figure 3.9.a, b and d) represent the same resource in different ways: the first one is an identifier, the second one is a Web page, and the last one is a document representation in N-triple format,

- **Problem 12 - IRI Coreference**: where two different IRIs are used to designate the same resource in the same way. Following the example of DBpedia in Figure 3.10, DBpedia uses different IRIs that provide information about resource "Luxembourg" in order to describe it. Also, DBpedia uses vocabularies for the predicates to connect the statements in a proper way. For example, `http://dbpedia.org/resource/Luxembourg` and `http://es.dbpedia.org/resource/Luxemburgo` (see Figure 3.10.a) identify the resource using conventional IRIs (cf. Section 2.2.4), while `http://schema.org/about` and `http://purl.org/dc/terms/subject` (see Figure 3.10.c) provide definitions using owner defining IRIs (cf. Section 2.2.4) to establish a concept definition for the predicate.

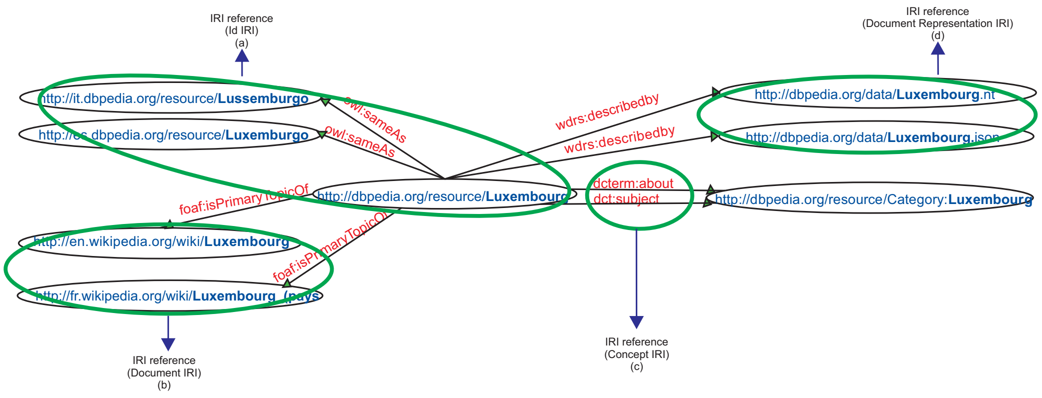

IRI reference
(Id IRI)
(a)
IRI reference
(Document Representation IRI)
(d)
http://it.dbpedia.org/resource/Lussemburgo
http://es.dbpedia.org/resource/Luxemburgo
owl:sameAs
owl:sameAs
wdrs:describedby
wdrs:describedby
http://dbpedia.org/data/Luxembourg.nt
http://dbpedia.org/data/Luxembourg.json
foaf:isPrimaryTopicOf
http://dbpedia.org/resource/Luxembourg
dcterm:about
dct:subject
http://dbpedia.org/resource/Category:Luxembourg
http://en.wikipedia.org/wiki/Luxembourg
foaf:isPrimaryTopicOf
http://fr.wikipedia.org/wiki/Luxembourg_(pays
IRI reference
(Document IRI)
(b)
IRI reference
(Concept IRI)
(c)

Figure 3.10: RDF Graph with IRI discrepancies - IRI reference

In short, various types of semantic ambiguities and IRI discrepancies can occur in an RDF description. For example, the fact that the same semantic information[1] can be described in totally different ways, can seriously complicate RDF data processing such as RDF indexing, storage, and querying (making it more difficult for example to define proper indexing structures based on syntactic cues, or formulate meaningful SPARQL queries). Furthermore, semantic ambiguities and IRI discrepancies in RDF may produce different kinds of logical redundancies (RDF graph-level) and physical (RDF serialization-level) disparities in the RDF descriptions which, on their own, can have a huge burden on RDF processing and the development of RDF databases and solutions (processing time, loading time, similarity measuring, mapping, alignment, and versioning) [Gea04, THTC$^+$15, THTCL16].

3.4.4 IRI Discrepancies creating Logical (Graph) Redundancies

Consider now the example given in Figure 3.11. Here, one can also identify various logical redundancies occurring in the forms of both RDF graph node duplications and edge duplications:

[1]Recall that the semantic information of an RDF statement refers not only to the values of the subject/predicate/object nodes/edges in the statement, but rather to the meaning of the statement as a whole: such that the meaning of a literal/blank/IRI node/edge depends on the subject/predicate/object nodes/edges it connects with in the containing statement.

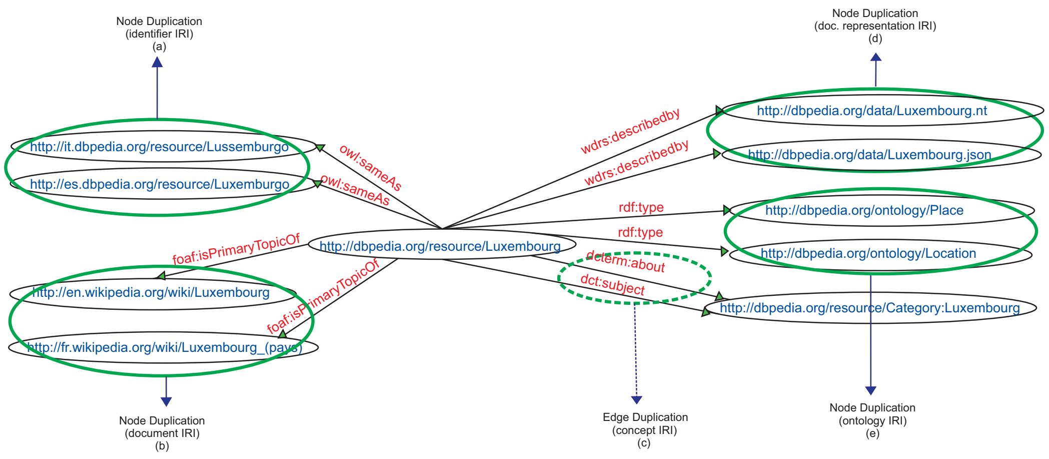

Node Duplication (identifier IRI) (a)
Node Duplication (doc. representation IRI) (d)
http://it.dbpedia.org/resource/Lussemburgo
http://es.dbpedia.org/resource/Luxemburgo
owl:sameAs
owl:sameAs
wdrs:describedby
wdrs:describedby
http://dbpedia.org/data/Luxembourg.nt
http://dbpedia.org/data/Luxembourg.json
rdf:type
rdf:type
http://dbpedia.org/ontology/Place
http://dbpedia.org/ontology/Location
http://dbpedia.org/resource/Luxembourg
foaf:isPrimaryTopicOf
foaf:isPrimaryTopicOf
http://en.wikipedia.org/wiki/Luxembourg
http://fr.wikipedia.org/wiki/Luxembourg_(pays)
dcterm:about
dct:subject
http://dbpedia.org/resource/Category:Luxembourg
Node Duplication (document IRI) (b)
Edge Duplication (concept IRI) (c)
Node Duplication (ontology IRI) (e)

Figure 3.11: RDF Graph with Logical redundancies due to IRI discrepancies (concerning problems 3 and 4)

- **Problem 13 - Node Duplication based on IRI discrepancies:** where equivalent IRI nodes, designating equivalent subjects and/or objects, appear more than once. For instance, Figures 3.11.a, b, d, and e highlight different node duplications with: *identifier IRI, document IRI, document representation IRI, and ontology IRI respectively.*

- **Problem 14 - Edge Duplication based on IRI discrepancies:** where equivalent IRI edges, designating equivalent RDF predicates, appear more than once, such as in Figure 3.11.c with highlights an edge duplication with *concept IRI*.

3.4.5 IRI Discrepancies creating Physical (Serialization) Disparities

IRI discrepancies can also produce disparities at the RDF serialization level, namely producing duplicate namespaces in the same RDF file. More formally:

- **Problem 15 - Namespace Duplication based on IRI discrepancies:** where two different namespaces are used to designate the same vocabulary, e.g., in Figure 3.8: `http://schema.org/` and `http://purl.org/dc/terms/` point to the same vocabulary.

3.5 Summary

In this chapter, we present different use cases in order to illustrate different kinds of redundancies and disparities which can occur in RDF descriptions, in order to help shape the direction of our research. From these use cases, we identified 15 research challenges that we called problems, which broadly fall into our four levels: logical redundancies, physical disparities, semantic ambiguities and IRI discrepancies.

One can clearly realize the compound effect of missing the different kinds of RDF logical duplications and physical disparities which can result from the various problems of syntactic redundancies (Sections 3.1 and 3.2), semantic ambiguities (Section 3.3) and IRI discrepancies (Section 3.4), all of which represent same (syntactic) or equivalent (semantic or coreferenced) RDF information which needs to be normalized into unified and unambiguous statements.

Against this background, in the next chapter, we introduce our first contribution towards the first two levels of the challenges: logical redundancies and physical disparities in a syntactic evaluation for RDF Normalization. Consequently, we cover semantic and IRI discrepancies in the following chapters.

Chapter 4

Syntactic RDF Normalization

“ We are what we repeatedly do. Excellence, then, is not an act, but a habit.”

— Aristotle

The necessity of working with RDF descriptions is increasing in an exponential way nowadays as discussed in Section 2.4. The impact of the Semantic Web and Linked Data to users (persons, organizations, etc.) of the Web affects the development of more applications that use RDF descriptions to publish their information in different areas.

All the information processed by these applications may contain duplications, since the users obtain the data of different sources. These duplications are the base of our motivation for cleaning the RDF descriptions as we describe with our uses cases in Sections 3.1 and 3.2.

In this chapter, we present our Syntactic RDF normalization contribution. First, we begin by further explaining some definitions developed for our normalization process (Section 4.1.1). We then propose normalization functions and normalization operators (Sections 4.1.2 and 4.1.3) for facilitating the creation of our normalization rules (Section 4.3). Next, we discuss related work which has influenced our understanding and the design of our approach (Section 4.2). We consequently establish our normalization properties (Section 4.4) for validating our normalization rules that we use in our normalization process (Section 4.5). Finally, in Section 4.6, we conclude this chapter with a summary.

4.1 Preliminaries

We start this section by providing definitions describing the main concepts related to our normalization process.

4.1.1 Definitions

Definition **4.1 (Extended Statement *[st$^+$]*)** *An extended RDF statement is a more expressive representation of a statement (**st**), denoted as:* $\boldsymbol{st}^+ \!:< s', p', o' >$ *where:*

- *s': $< s, ts >$ is a tuple that we call "extended subject" composed of the subject value (s) and its type (ts), where $ts \in \{u, bn\}$.*
- *p': $< p, dt, lang >$ is a 3-tuple that we call "extended predicated" composed of the predicate value (p), its datatype $dt \in DT \cup \{\perp\}$, and language tag $lang \in Lang \cup \{\perp\}$. $\perp$ represents a "null" value.*
- *o': $< o, to >$ is a tuple that we call "extended object" composed of the object value (o) and its type $to \in \{u, bn, l\}$⧫*

Recall that: u stands for IRI, bn stands for blank node, and l stands for literal.

The following notation is adopted to represent an extended statement:

$$st^+ :< s_{ts\in\{u,bn\}}, p_{dt}^{lang}, o_{to\in\{u,bn,l\}} >$$

Based on the example of Figure 2.7, RDF statement st_1 becomes extended statement $st_1^+ =$ <`http://www.univ-pau.fr`$_u$, ex:nameProf$_\perp^\perp$, bn$_1_{bn}$ >.

The function $ST^+(G)$ will be used in the following to return all the extended statements of an RDF description.

In the rest of the study, we use the extended statement definition to represent RDF statements in the normalization process.

Definition **4.2 (RDF Description *[D]*)** *An RDF description D stands for either: i) an RDF graph G (when referring to the RDF logical representation), or ii) an RDF file F (when referring to the RDF physical representation), depending on the context and application at hand*⧫

Definition **4.3 (RDF Element *[E]*)** *An RDF element E refers to any of the components of an RDF description, including either: i) IRI, ii) literal, iii) blank node, vi) statement, v) extended statement, vi) namespace, or vii) prefix, depending on the context and application at hand*⧫

Definition **4.4 (RDF Normalization)** *RDF normalization is the process that transforms a (non-normalized) RDF description noted D (i.e., either an RDF graph G or an RDF file F) into another (normalized) RDF description noted $\tilde{D}$ = Norm(D) (i.e., either a normalized RDF graph $\tilde{G}$ or a normalized RDF file $\tilde{F}$), which is equivalent in its semantic expressiveness to D after eliminating logical redundancies and physical disparities*[1]*, and establishing an appropriate order (indexing [WKB08]) of the resulting serialized RDF statements [THTCL16, THTC+15]*♦

Note that in our definition or normalization, we consider the ordering of statements within the serialization of an RDF description (i.e., the RDF file) as an important issue because it can affect the resulting RDF description's processing efficiency within the target application, w.r.t. storage, similarity, processing time, querying, etc., which we further discuss in Sections 4.5.1 and 6.4.3.

In order to fulfil our RDF normalization process, we first develop a set of dedicated functions (Section 4.1.2) and operators (Section 4.1.3), which will serve as "construction tools", utilized to formulate/build a set of formal normalization rules (Section 4.3) with provable properties comprising the main pillars of our framework (Section 4.5).

4.1.2 Normalization Functions

We develop several functions to be utilized in defining our normalization rules. These can be categorized in two main groups: i) basic functions which are related to the basic RDF model detailed in Section 2.3.1 (cf. Table 4.1), and ii) normalization functions which are defined to serve our concept of RDF normalization (cf. Table 4.2).

i) Basic Functions:

In the remainder, the following functions ***R***, ***U***, ***L***, ***BN***, ***ST***$^+$, ***NS***, ***Px***, ***Enc***, and ***QN*** will be used respectively to return all the Resources (IRIs and literals), IRIs, literals, blank nodes, Extended Statements, Namespaces, Prefixes, Encoding, or Qnames of an RDF description *D*. Additionally, we created other basic function named ***UNS*** to return all the Unused Namespaces of an RDF description *D*.

[1]While our definition of RDF normalization is comparable to the notion of RDF *normal form* in [Gea11], nonetheless the authors in [Gea11] mostly address the normalization of RDF statements defined using the RDFS vocabulary and disregard the kinds of redundancies and disparities addressed in this study. Regarding simple RDF, the authors in [Gea11] aim to eliminate blank nodes in defining *lean graphs* in order to produce *normal forms* later (cf. Section 4.2.2). In this context, our approach completes the study in [Gea11] by eliminating redundancies from the graph, which can then be processed to produce a *lean* and *normal form* representation following [Gea11]. Recall that the study in [Gea11] does not address physical (serialization) disparities.

[1]Function names here were specifically chosen to match their outputs' names (in most cases) in order to simplify the description of normalization operators, rules, and properties later on (cf. Sections 4.3 and 4.5).

Table 4.1: Summarized descriptions of functions based on the basic definitions of RDF

Function [1]	Input	Output	Description
ST^+	$G\|F$	ST^+	Returns all the extended Statements of an RDF Graph
U	ST^+	U	Returns a set of IRIs from ST^+
L	ST^+	L	Returns a set of literals from ST^+
BN	ST^+	BN	Returns a set of blank nodes from ST^+
R	ST^+	R	Returns a set of Resources from ST^+
Enc	ST^+, enc	F	Returns the encoding of an RDF file based on the parameter $enc \in \{RDF/XML, Turtle, etc.\}$
QN	F	QN	Returns all the QNames of an RDF file
NS	F	NS	Returns all the Namespaces of an RDF file
Px	F	Px	Returns all the Prefixes of an RDF file
UNS	F	UNS	Returns all the Unused Namespaces of an RDF file

ii) Normalization Functions:

In this section, we develop a group of normalization functions to handle specific characteristics of the extended RDF statements, and which will be utilized in defining our normalization rules, including: identifying statements outgoing from blank nodes, computing the cardinality of RDF descriptions based on the number of extended statements, removing or replacing elements in extended statements, or changing the order of extended statements in RDF descriptions.

FUNCTION **1 (Extended Statement Outgoings *[O]*)** *The extended statement outgoings function accepts as input an RDF extended statement st_i^+ and returns as output the set of all the extended statements deriving from the input extended statement. More formally, the outgoings of st_i^+, noted $O(st_i^+)$, designate the set of extended statements having for subject the object element o_i' of st_i^+:*

$$O(st_i^+) = \{st_j^+, ..., st_n^+\}$$

where:

$$st_j^+ =< (o_i', p_j', o_j' >, ..., st_n^+ =< o_i', p_n', o_n' > \blacklozenge$$

For example, considering extended statement st_i^+ in Figure 4.1, we identify the following outgoings of st_i^+:

$$O(st_1^+) = \{st_5^+, st_6^+\}$$

$$O(st_1^+) = \{< bn\_1_{bn}, p2_{string}^{\perp}, l_{1l} >, < bn\_1_{bn}, p3_{\perp}^{fr}, l_{2l} >\}$$

Note that for the sake of simplicity, we only define extended statement outgoings here, and not extended statement incomings (which would underline triples incoming into the extended statement's objects) since the latter are not specifically useful in the syntactic normalization process.

FUNCTION **2 (RDF Description Cardinality** *[||]*) *The RDF Description Cardinality function, noted card(D) or $|D|$ for short, accepts as input an RDF description D, and returns as output the number of extended statements in D, card(D) = $|ST^{+}|$*♦

For instance, in Figure 4.2, $|G| = 9$ since the cardinality of G takes into account all the extended statements.

FUNCTION **3 (RDF Description Minimal Cardinality** *[||||]*) *The RDF Description Minimal Cardinality function, noted cardMin(D) or $\|D\|$ for short, takes as input an RDF description D (where D is either an RDF graph G or an RDF file F), and returns as output the number of distinct extended statements in D while disregarding duplicated statements, cardMin(D) = $\|ST^{+}\|$*♦

For instance, in Figure 4.2, $\|G\| = 4$ since the cardinality of G does not take into account the two duplicated extended statements:
$< st_3^{+} >:< u_{1u}, p4_{\perp}^{\perp}, u_{2u} >$,
$< st_8^{+} >:< bn\_2_{bn}, p3_{\perp}^{\perp}, l_{2l} >$, and
all the statements linked with bn_1 because bn_1 is contained in[1] bn_2

FUNCTION **4 (Remove RDF element** *[remove]*) *The remove RDF element function noted: remove(e, D), accepts as input an RDF description D (or any of its components, e.g., $ST^{+}(D)$, $QN(D)$, etc.) and an RDF element e (i.e., an RDF statement or any of its components, e.g., IRI, literal, iii) blank node, etc., cf. Definition 4.3), and returns as output a variation D' of D, where an element e has been removed, i.e., remove (e, D) = D' = D - e. Note that the remove function is mainly designed to remove duplicate statements and/or unused Qnames (cf. Section 4.3)*♦

For instance, in Figure 4.2 given $st_3^{+} = st_4^{+}$, we can apply $remove(st_4^{+}, ST^{+}(G))$ to delete the extended statement duplication, because both statements are equals.

FUNCTION **5 (Replace RDF element** *[replace]*) *The replace RDF element function noted: replace(i, j, D), accepts as input an RDF description D (or any of its components, e.g.,*

[1]The operator of "contained in" is presented in Section 4.1.3.

$ST^+(D)$, $QN(D)$, etc.), as well as two qname prefixes: i and j, and returns a variation D' of D, where all occurrences of i have been replaced by j♦

For instance, in Figure 3.3, there are two prefixes ex and $ex1$ that are equivalent (i.e., $ex = ex1$) because they refer to the same namespace "`http://example.org/stuff/1.0/`". As a result, applying $replace(ex1, ex, ST^+(F))$ returns a variation of RDF file F where all occurrences of *ex1* have been replaced by *ex.*

Function **6 (Order RDF Statements** *[order]*) *The order RDF statements function, noted: order(st_i^+, st_j^+, D, $\tilde{p}$), accepts as input an RDF description D and two extended statements st_i^+ and st_j^+ within D, and an ordering parameter $\tilde{p}$, and returns a variation D' of D, where st_i^+ and st_j^+ have been ordered following the ordering parameter $\tilde{p}$, i.e., $st_i^+ <_{\tilde{p}} st_j^+$, where $<_{\tilde{p}}$ allow us to order from lowest to highest statements according to the parameter $\tilde{p}$. Parameter $\tilde{p}$ highlights the user's preference in ordering statements (which can be done following their subjects, predicates, objects, or their combination, described in details in Section 4.5.2)*♦

Table 4.2: Summarized descriptions of functions based on definition of the normalization process

Function	Input	Output	Description
O	st^+	$O(st^+)$	Returns all the outgoings of st^+
$card$	D	$\|D\|$	Returns the number of elements of D including duplications
$cardMin$	D	$\|\|D\|\|$	Returns the number of elements of D without duplications
$remove$	i, D	$D - i$	Returns $D \in \{G, F\}$ without element(s) $i \in \{ST^+(D), QN(D)\}$
$replace$	$i, j, ST^+(D)$	$ST^+(D)$	Returns all the statements $ST^+(D)$ updated with the replacement of j by i where $i \wedge j \in QN(D)$
$order$	st_i^+, st_j^+, D, $\tilde{p}$	$ST^+(D)$	Returns ST^+ of D ordered following the user's preference order based on parameter $\tilde{p}$

4.1.3 Normalization Operators

Here, we make use of the functions introduced in the previous section to define a set of operators (see Table 4.3) needed to represent the *equality* relationship between RDF statements, graphs, and files, which will be chiefly utilized in identifying (and then eliminating) duplicate statements in our normalization process (i.e., informally, we need to eliminate all repetitive occurrences of equal - and thus redundant - RDF statements).

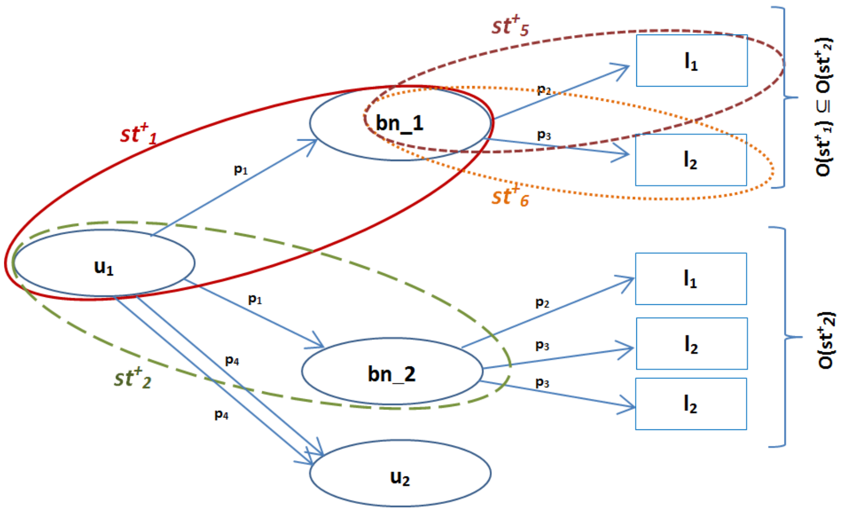


Figure 4.1: RDF example of extended statement containment depicting the graph in Figure 3.1 where st_i^+, u_i, p_i, bn_i, and l_i respectively represent corresponding extended statements, IRIs, predicates, blank nodes, and literals.

Operator **1 (Statement Containment** *[$\preceq$]***)** *An extended statement st_i^+ is said to be contained in another extended statement st_j^+, noted $st_i^+ \preceq st_j^+$, if: 1) all the outgoings of st_i^+ occur in st_j^+, i.e.,* ***$O(st_i^+) \subseteq O(st_j^+)$****, 2) both st_i^+ and st_j^+ have the same subject and predicate, and 3) the object type (to) in both statements is a blank node (bn). Formally:*

$$\forall st_i^+, st_j^+ \in G, st_i^+ \preceq st_j^+ \iff st_i^+.to = \text{“}bn\text{”} \wedge st_j^+.to = \text{“}bn\text{”} \wedge st_i^+.s = st_j^+.s \wedge st_i^+.p = st_j^+.p \wedge (O(st_i^+) \subseteq O(st_j^+))\blacklozenge$$

For instance, in Figure 4.1, $st_1^+ \preceq st_2^+$ since they share the same subject (i.e., u_1) and the same predicate (i.e., p_1), and have O(st_1^+) $\subseteq$ O(st_2^+) w.r.t their outgoings.

Similarly, we say an extended statement st_i^+ is not contained in another extended statement st_j^+ (i.e., $st_i^+ \npreceq st_j^+$) if any of the extended statements containment conditions (related to the subject, predicate, and outgoings, defined with Operator 1)) does not hold. Note that blank node identifiers (when available) do not affect the statement containment property.

The statement containment property will be mainly useful in detecting node duplications (cf. Section 4.3 and Section 4.5).

Operator **2 (Extended Statement Equality** *[$=_{st}$]***)** *An extended statement st_i^+ is said to be equal to another extended statement st_j^+, noted $st_i^+ =_{st} st_j^+$, if and only if: 1) the subject of st_i^+ is equal to the subject of st_j^+, 2) the predicate of st_i^+ is equal to the predicate of st_j^+, and 3) the object of st_i^+ is equal to the object of st_j^+. Formally:*

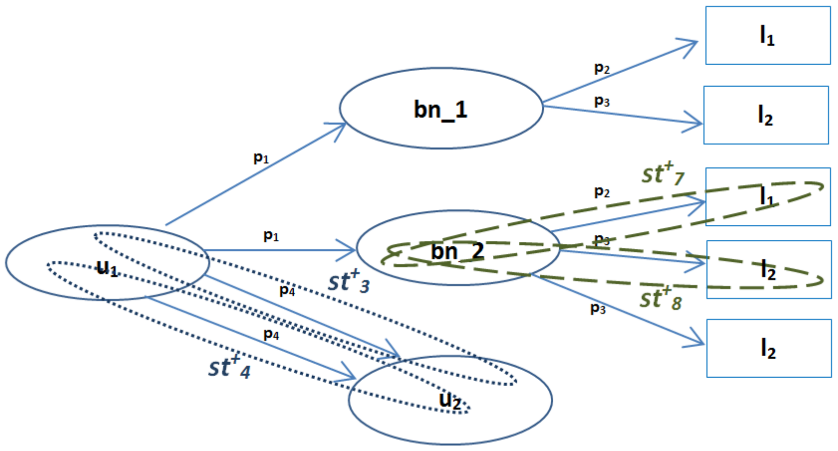


Figure 4.2: RDF example with equal and different extended statements.

$$\forall st_i^+, st_j^+ \in G, st_i^+ =_{st} st_j^+ \iff st_i^+.s_i' = st_j^+.s_j' \wedge$$
$$st_i^+.p_i' = st_j^+.p_j' \wedge st_i^+.o_i' = st_j^+.o_j'$$

if:

$$st_i^+.s_i' = st_j^+.s_j' \iff st_i^+.s_i = st_j^+.s_j \wedge st_i^+.ts_i = st_j^+.ts_j \wedge$$
$$st_i^+.p_i' = st_j^+.p_j' \iff st_i^+.p_i = st_j^+.p_j \wedge st_i^+.dt_i = st_j^+.dt_j \wedge st_i^+.lang_i = st_j^+.lang_j \wedge$$
$$st_i^+.o_i' = st_j^+.o_j' \iff st_i^+.o_i = st_j^+.o_j \wedge st_i^+.to_i = st_j^+.to_j \blacklozenge$$

For instance, in Figure 4.2, $st_3^+ =_{st} st_4^+$ since they share the same subject (i.e., u_1), the same predicate (i.e., p_4), and the same object (i.e., u_2).

Similarly, we say an extended statement st_i^+ is not equal (i.e., unequal) to another extended statement st_j^+ (i.e., $st_i^+ \neq_{st} st_j^+$) if any of the extended statements equality conditions (related to the subject, predicated, and object, defined with Operator 2)) does not hold. Formally:

$$\forall st_i^+, st_j^+ \in G, st_i^+ \neq_{st} st_j^+ \iff st_i^+.s_i' \neq st_j^+.s_j' \vee$$
$$st_i^+.p_i' \neq st_j^+.p_j' \vee st_i^+.o_i' \neq st_j^+.o_j'$$

For instance, in Figure 4.2, $st_7^+ \neq_{st} st_8^+$ since they share the same subject (i.e., u_1) but with different predicates (i.e., p_2 and p_3) and different objects (i.e., l_1 and l_2).

Operator **3 (Extended Statement Intersection** *[∩]*) *The extended statement set ST_i^+ of an RDF Description D_i, i.e., $ST_i^+(D_i)$, is said to intersect with another extended statement* *set ST_j^+ of an RDF Description D_j, i.e., $ST_j^+(D_j)$, if and only if there exists an extended statement st^+ that simultaneously belongs to two extended statement sets: $ST_i^+(D_i)$ and $ST_j^+(D_j)$. Formally:*

$$ST_i^+(D_i) \cap ST_j^+(D_j) = \{st^+ / st^+ \in ST_i^+(D_i) \wedge st^+ \in ST_j^+(D_j)\}$$

For instance, regarding Figure 4.2 and Figure 4.5 as two RDF graphs G_i and G_j respectively, then $ST_i^+(G_i) \cap ST_j^+(G_j) = \{(u_1, p_1, bn\_1), (u_1, p_4, u_2), (bn\_1, p_2, l_1), (bn\_1, p_3, l_2)\}$

Operator **4 (RDF graph Equality** *[$=_{RDFG}$]*) *An RDF graph G_i is said to be equal to another RDF graph G_j, noted $G_i =_{RDFG} G_j$, if and only if: i) all the extended statements of G_i are equal or contained in extended statements of G_j and vice versa, and ii) they have equal minimum cardinalities. Formally:*

$$G_i =_{RDFG} G_j$$

if:

$$\begin{gathered} \forall st_i^+ \in G_i, \exists st_j^+ \in G_j / st_i^+ =_{st} st_j^+ \vee st_i^+ \preceq st_j^+ \wedge \\ \forall st_j^+ \in G_j, \exists st_i^+ \in G_i / st_j^+ =_{st} st_i^+ \vee st_j^+ \preceq st_i^+ \wedge \\ \|G_i\| = \|G_j\| \blacklozenge \end{gathered}$$

In other words, $G_i =_{RDFG} G_j$ means that both graphs share the same extended statements, and thus the same semantic expressiveness; without necessarily being normalized, i.e., they can contain logical redundancies (duplicate statements).

Operator **5 (RDF file Equality** *[$=_{RDFF}$]*) *An RDF file F_i is said to be equal to another RDF file F_j, noted $F_i =_{RDFF} F_j$, if and only if: i) their corresponding RDF graphs are equal, i.e., $G_i =_{RDFG} G_j$, ii) their corresponding namespaces are equal, i.e., $NS(F_i) = NS(F_j)$ and iii) F_i is serialized following the same encoding format (i.e., enc_i) as F_j (i.e., enc_j). Formally:*

$$F_i =_{RDFF} F_j$$

if:

$$\begin{gathered} G_i =_{RDFG} G_j \wedge \\ NS(F_i) = NS(F_j) \wedge \\ Enc(F_i, enc_i) \wedge Enc(F_j, enc_j) \wedge enc_i = enc_j \blacklozenge \end{gathered}$$

In other words, $F_i =_{RDFF} F_j$ means both files share equivalent logical representations (equal RDF graphs), they share the same namespaces, and they are serialized using the same encoding format.

Table 4.3: Summarized descriptions of operators based on definition of the normalization process

Operator	Description
$\preceq$	Containment between two extended statements when the objects are blank nodes
$=_{st}$	Equality between two extended statements
$\cap$	Intersection between two extended statements
$=_{RDFG}$	Equality between two RDF graphs
$=_{RDFF}$	Equality between two RDF files

4.2 Related Work

The need for RDF normalization has been identified and discussed in various domains, ranging over domain-specific knowledge representation and data integration. Yet, few existing studies have specifically addressed the issues of logical (graph) and physical (syntax) RDF normalization.

In the following, for clarity of presentation, we review RDF normalization approaches based on the application (knowledge representation and data integration) and the evaluation level they use for their elements (i.e., logical and physical). Therefore, we classify the methods in three categories: i) Knowledge Representation and Integration, ii) RDF graph normalization, and iii) RDF syntax normalization. We estimate that this categorization provides the simplest and most consistent unified view of the wide variety of diverse approaches proposed for the literature.

The kinds of RDF data being treated as well as the limitations and intended applications domains will be discussed for each approach. Catalogs summarizing the properties and characteristics of all covered approaches are depicted in Tables 4.4 and 4.5.

4.2.1 Knowledge Representation and Integration

Various approaches have been developed to normalize knowledge representation in RDF, namely in the bioinformatics domain [Tea09, Pea09, Jea13, Bea08, Nea12]. In [Tea09], the authors provide an approach to map LexGrid [Pea09], a distributed network of lexical resources for storing, representing and querying biomedical ontologies and vocabularies, to various Semantic Web (SW) standards, namely RDF and SKOS[1]. They introduce the LexRDF project which leverages LexGrid, mapping its concepts and properties to standard (normalized) RDF tagging

[1]The Simple Knowledge Organization System (SKOS) is a stand-alone vocabulary, built with OWL and RDFS, designed to create controlled vocabularies and thesauri in RDF.

following the SKOS [BM09] specification, thus providing a unified RDF based model (using a common terminology) for both semantic and lexical information describing biomedical data. In [Jea13], the authors introduce a framework designed to allow open data access and collaboration for ICD-11 (International Classification of Diseases, version 11[1]). The RDF normalization process developed in this approach includes: i) generating uniform IDs for ICD-11 categories using the ICD URI scheme[2] proposed by WHO (World Health Organization), and ii) normalizing lexical properties of ICD-11 contents using the SKOS RDF model. In a related study [Nea12], the authors introduce the Bio2RDF project, aiming to create a network of coherent linked data across life sciences databases. The authors address URI normalization, as a necessary prerequisite to build an integrated bioinformatics data warehouse on the SW, where resources are assigned URIs normalized around the bio2rdf.org namespace. Table 4.4 shows the summarization of all the approaches presented in this section.

4.2.2 RDF Graph (Logical) Normalization

While various studies have highlighted the need for RDF normalization, yet very few have actually targeted the issues of RDF logical (graph) and physical (syntactic/serialization) normalization. In [HG04], Hayes and Gutierrez target RDF graph model normalization. The authors argue that the notion of RDF graph has not been explicitly defined in the RDF specification [KC04], it does not distinguish clearly among the term "RDF Graph", the mathematical concept of graph, and the graph-like visualization of RDF data. The authors discuss some of the redundancies which can occur in a traditional RDF directed labeled graph (cf. Section 4.1.1), particularly regarding the connectivity of resources. Namely, an RDF graph edge label (i.e., a predicate) can occur redundantly as the subject or the object of another statement (e.g., $< dbpedia : Researcher, dbpedia : Workplace, dbpedia : University >$ and $< dbpedia : Workplace, rdf : type, dbpedia : Professional >$). Hence, the authors in [HG04] introduce an RDF graph model as a special bipartite graph where RDF triples are represented as ordered 3-uniform hypergraphs where edge nodes correspond to the $< subject, predicate, object >$ triplet constituents, ordered following the statement's logical triplet ordering. The new model is proven effective in reducing the predicate-node duplication redundancies identified by the authors.

In subsequent studies [Gea04, Gea11], the authors address the problem of producing RDF normal forms and evaluating the equivalence among them. The studies in [Gea04, Gea11] specifically target the RDFS vocabulary with a set of reserved words to describe the relationships

[1] `http://www.who.int/classifications/icd/en`

[2] The ICD URI scheme is dedicated for naming and supporting Web Services handling ICD data. It is available on `http://id.who.int`, with `http://id.who.int/icd/schema` as the prefix for the vocabulary terms that relate to the ICD classification, and `http://id.who.int/icd/entity` for the fundamental foundation entities related to ICD concepts.

Table 4.4: Summarized knowledge representation and integration approaches

App.	**Data Targeted**	**Features**	**Limitations**	**Aplication**		**Output**
				Domain	**Area**	
Pathak et al. [Pea09]	OWL, RRF, OBO, XML, Text	Proposal name: LexGrid • Identifying logical inconsistencies in ontologies and vocabularies • Providing a consistent standardized rich API to access multiple vocabularies and ontologies distribution • Giving a standard storage of controlled vocabularies and ontologies	Blank nodes Literals Statements	Biomedical	Querying	LexGrid
Tao et al. [Tea09]	LexGrid	Proposal name: LexRDF project • LexGrid [Pea09] data is considered to be properly described • Mapping the concepts and properties to standard - normalized RDF tagging • Following SKOS specification • Considering the normalization as a result between the correctly mapping of LexGrid to LexRDF	Blank_nodes Literals Statements	Biomedical	Querying and Storing	RDF triples
Jiang et al. [Jea13]	RDF triples (ICD-11)	Framework designed to allow open data access and collaboration • Mapping data ICD-11 alpha to the new information • RDF normalization process: (a) Generating uniform IDs using ICD URI scheme, and (b) Normalizing lexical properties of ICD-11 using the SKOS RDF model	Blank_nodes Literals Statements Namespaces	Biomedical	Knowledge Representation	JSON XML RDF/XML Turtle
Belleau et al. [Bea08]	RDF triples	Proposal name: Bio2RDF project • URI normalization • Producing Bio2RDF statements • Probably done manually by domain experts	Blank_nodes Literals Statements Namespaces	Biomedical	Data Integration	RDF triples

between resources (e.g., rdfs:type, rdfs:range, rdfs:domain, etc.). They provide a full-fledged theoretical model including notions such as:

i) ***RDF lean graph*** as a minimal graph preserving all URIs of its origin graph while having fewer blank nodes (where minimality designates that the RDF graph cannot be further reduced), and

ii) ***RDF normal form*** as a unique representation of an RDF lean graph (where uniqueness designates that the lean RDF graph is unique with respect to the original RDF graph).

First, the authors do not identify nor target the kinds of redundancies and disparities addressed in our study, both at the logical and physical levels (e.g., edge/node duplications, literals, IRIs, prefixes, and data-types, among others). Second, they focus on RDFS vocabulary constructs which are out of the scope of our study. Third, the authors' main motivation in [Gea11] is different from ours: they aim to reduce (simplify) RDF query answers by: i) producing the answer, and then ii) generating its normal form. They discuss RDF query language features and how those should translate to process the logical RDF descriptions (graphs). In contrast, our study aims at normalizing (simplifying) RDF descriptions from the start, independently of any particular application, targeting both logical (graph) and physical (serialization) levels, so that querying (and other applications/functionality) can be later performed on the normalized data.

In short, while the studies in [Gea04, Gea11, HG04] thoroughly cover general theoretical foundations of RDF logical (graph) representation and processing, our approach completes the latter by targeting specific logical (graph) redundancies (and physical/serialization disparities) which were out of the scope of [Gea04, Gea11, HG04], namely distinct edge (predicate) duplication, node (subject/object) duplication, and combined edge and node (whole statement) duplication, as well as all kinds of physical disparities (see motivation Section 3).

In a related study [Fea13], the authors introduce the binary RDF representation for publication and exchange called: HDT (Header-Dictionary-Triples) serialization format. The HDT representation format is based on three main components: i) a Header that includes metadata describing the RDF dataset, ii) a Dictionary that organizes all the identifiers in the RDF graph, and iii) Triples which represent the pure structure of the underlying RDF graph. Using this format, the authors reduce the verbosity/redundancy and storage space of the RDF files while transforming blank nodes into IRIs, thus losing the meaning of the blank node in defining RDF statements.

Recent approaches [SL13, Lon15] introduce a graph normalization algorithm which extend toward RDF dataset normalization. In [SL13], the authors transform an RDF graph into a standard form, generating a cryptographically-strong hash identifier for the graph, or digitally signing it. The authors define normalization as *"the process of taking an input graph and performing a transformation on that input that results in all the aspects of the graph being arranged in a deterministic way in the output graph"*. In [Lon15], the author extends the RDF normalization approach from [SL13] toward so-called RDF dataset normalization, revising the concept of normalization as *"the process of transforming an input RDF dataset to a normalized RDF dataset. That is, any two input RDF datasets that contain the same information, regardless of their arrangement, will be transformed into identical normalized RDF datasets"*. The proposed algorithms in [SL13, Lon15] take a JSON-LD input format, and provide an output in N-triple serialization while relabeling certain nodes and erasing certain redundancies.

Yet, the authors in [Fea13, SL13, Lon15] do not control redundancies within RDF graphs containing blank nodes, and do not address serialization disparities.

4.2.3 RDF Syntax (Physical) Normalization

At the physical (syntactic) level, Vrandecic et al. [Vea09] argue that the same RDF graph can be expressed in many different ways in RDF/XML serialization, using different RDF constructs, thus complicating the processing of RDF descriptions. The authors introduce a method to normalize the serialization of an RDF graph using XML grammar (DTD) definitions. The process consists of two steps: (a) Defining an XML grammar (DTD) with whom all generated RDF/XML serializations should comply; the DTD is generated semi-automatically, such that the system provides a tool box to help the user (expert) to choose elements and attributes/properties following her serialization needs and, (b) Defining SPARQL query statements to query the RDF dataset in order to return results, consisting of serializations compliant with the grammar (DTD) at hand. This is comparable to the concept of semantic mediation using SPARQL queries [Kea08]. Note that SPARQL statements are automatically generated based on the grammar (DTD). The authors provide an online implementation[1] to demonstrate the usefulness of their proposal. Here, we note that the authors' motivation in [Vea09] clearly corresponds to the same problem addressed in our proposal. Nonetheless, we consider serialization disparities as well as logical (blank node) redundancies which are not addressed in the mentioned work. In other words, our approach is complementary to the method in [Vea09].

To sum up, our approach completes and builds on existing methods to normalize RDF information, namely [Gea04, HG04, Vea09, Gea11, SL13, Lon15], by handling logical and physical redundancies and disparities which were (partially or totally) unaddressed in the latter.

Table 4.5 depicts the summarization of RDF Graph (Logical) and Syntax (Physical) Normalization approaches developed in the literature where the approaches have several limitation w.r.t. blank nodes, literals, URI, namespaces and statements, that we overcome in our

[1]`http://km.aifb.kit.edu/services/RDFSerializer/`

approach.

4.3 Normalization Rules

In this section, we provide a set of rules that allow to solve the motivation problems in Section 3. We consequently establish two normalization goals here: i) solving logical redundancies (discussed in Section 3.1) and ii) solving physical disparities (discussed in Section 3.2).

In the following, we use $\tilde{D}$, $\tilde{G}$, and $\tilde{F}$ to designate a normalized RDF description, RDF graph, and RDF file respectively (cf. Definition 4.4).

4.3.1 Solving Logical Redundancies

Logical redundancies related to node duplication, edge duplication, and node/edge duplications (presented in Section 3.1) can be eliminated from an RDF graph G by applying the following transformation rules:

- **Rule 1 - Statement Equality Elimination (R1)**: It is designed to eliminate edge duplications and/or node duplications within individual extended statements. More formally:

  $\forall st_i^+, st_j^+ \in ST^+(G)$ / $i \neq j$, if $st_i^+ =_{st} st_j^+ \Longrightarrow remove(\{st_j^+\}, ST^+(G))$

  Given two equal extended RDF statements st_i^+ and st_j^+ in an RDF graph G such that $st_i^+ =_{st} st_j^+$, applying Rule 1 on G produces another RDF Graph G' where st_j^+ has been removed⧫

Lemma 1 *Given two extended statements st_i^+, $st_j^+ \in ST^+(G)$ where $st_i^+ =_{st} st_j^+$, applying Rule 1 on G produces an RDF Graph G' verifying at least one of the following features:*

- $ST^+(G') \subseteq ST^+(G)$ / $|ST^+(G')| = |ST^+(G)| - 1$ *(reducing the number of edge duplications by 1 where the object of the statement is represented either as a IRI or as a literal);*
- $L(G') \subseteq L(G)$ / $|L(G')| = |L(G)| - 1$ *(reducing the number of node duplications by 1 where the object of the statement is represented as a literal).*

Table 4.5: Summarized RDF Graph (Logical) and Syntax (Physical) Normalization approaches

App.	**Data Targeted**	**Order**[a]	**Exploited RDF Elements**	**Features**	**Limitations**	**Aplication**		**Output**
						Domain	**Area**	
RDF Graph (Logical) Normalization								
Hayes et al. [HG04]	RDF Graph	SPO	Statement	Proposal name: Bipartite RDF graph • Reducing redundancies (edge - node duplication) • Improving the connectivity between resources • Better distinction between schema and data statement	Blank nodes Literals URIs Namespaces	Not specified	Storage and Clustering	Bipartite Graph
Gutierrez et al. [Gea04, Gea11]	RDF Graph	Not considered	Blank nodes Resources	• Formalizing minimal and maximal representations • Establishing normal forms for RDF data	Statements Namespaces	RDFs vocabularies	Querying	RDF Graph
Longley [Lon15]	JSON-LD	Alphab. based on N-triple	Blank nodes Resources	• Transforming an RDF graph into a standard form • Generating a cryptographically âĂŞ strong hash identifier for the graph • Relabeling certain nodes and erasing certain redundancies	Statements Namespaces	Not specified	Linked Data	N-triple
Fernandez et al. [Fea13]	N-triples	SPO	Resources	Proposal name: RDF HDT format • Reducing verbosity using three elements: Header, Dictionary, triples	Blank nodes Statements Namespaces	Not specified	Data Management and Compression	HDT
RDF Syntax (Physical) Normalization								
Vrandecic et al. [Vea09]	Arbitrary RDF âĂŞ file	Not considered	Statement	• Defining XML grammar (DTD) to generate RDF/XML serialization • Defining SPARQL query statements to query the RDF dataset -> comparable to the concept of semantic mediation in [Kea08]	Blank nodes Literals URIs Namespaces	FOAF vocabulary	Querying	XSLT

[a]Designates the order of statements in the RDF description, which can be performed following the subject (S), predicate (P), and/or object (O) elements of the statements (cf. details in Section 4.5.2).

PROOF **1** *Given two extended statements st_i^+, $st_j^+ \in ST^+(G)$ where $st_i^+ =_{st} st_j^+$, applying Rule 1 on G produces an RDF graph G' which is identical to G except that in G': the redundant extended statement st_j^+ has been removed. This means that:*

- *When $st_j^+.to$ = "l" $\vee$ $st_j^+.to$ = "u", the set of extended statements in the resulting graph G' is included in that of G, i.e., $ST^+(G') \subseteq ST^+(G)$ such that $ST^+(G') = ST^+(G) - \{st_j^+\}$ since exactly one extended statement st_j^+ has been removed, which means $|ST^+(G')| = |ST^+(G)| - 1$.*
- *When $st_j^+.to$ = "l", the set of literals in the resulting graph G' is included in that of G, i.e., $L(G') \subseteq L(G)$ such that $L(G') = L(G) - \{st_j^+.o\}$ since exactly one literal value $st_j^+.o$ has been removed, which means $|L(G')| = |L(G)| - 1$.*

LEMMA **2** *Given two subsets $ST_i^+(G), ST_j^+(G) \subset ST^+(G)$ where $\forall st_i^+ \in ST_i^+(G) \wedge st_j^+ \in ST_j^+(G)$ / $st_i^+ =_{st} st_j^+$, applying Rule 1 on G produces an RDF Graph G' verifying at least one of the following features:*

- $ST^+(G') \subseteq ST^+(G)$ / $|ST^+(G')| = |ST^+(G)| - |ST_i^+(G) \cap ST_j^+(G)|$ *(reducing the number of edge duplications by $|ST_i^+(G) \cap ST_j^+(G)|$ where the object of the statement is represented either as a IRI or as a literal);*
- $L(G') \subseteq L(G)$ / $|L(G')| = |L(G)| - |L_i(G) \cap L_j(G)|$ *(reducing the number of node duplications by $|L_i(G) \cap L_j(G)|$ where the object of the statement is represented as a literal).*

PROOF **2** *Given two extended statements $ST_i^+(G), ST_j^+(G) \subset ST^+(G)$ where $\forall st_i^+ \in ST_i^+(G) \wedge st_j^+ \in ST_j^+(G)$ / $st_i^+ =_{st} st_j^+$, applying Rule 1 on G produces an RDF graph G' which is identical to G except that in G': redundant extended statements in $ST_i^+(G) \cap ST_j^+(G)$ have been removed. This means that:*

- *When ($\forall st^+ \in (ST_i^+(G) \cap ST_j^+(G))$ / $st^+.to$ = "l" $\vee$ $st^+.to$ = "u"), the set of extended statements in the resulting graph G' is included in that of G, i.e., $ST^+(G') \subseteq ST^+(G)$ such that $ST^+(G') = ST^+(G) - (ST_i^+(G) \cap ST_j^+(G))$ since extended statements duplicated in $ST_i^+(G) \cap ST_j^+(G)$ have been removed, which means $|ST^+(G')| = |ST^+(G)| - |ST_i^+(G) \cap ST_j^+(G)|$.*
- *When $\forall st^+ \in (ST_i^+(G) \cap ST_j^+(G))$ / $st^+.to$ = "l", the set of literals in the resulting graph G' is included in that of G, i.e., $L(G') \subseteq L(G)$ such that $L(G') = L(G) - (L_i(G) \cap L_j(G))$ since $L_i(G) \cap L_j(G)$ literals, which are the objects of the extended statements where $L_i(G) \cap L_j(G)$, have been removed, which means $|L(G')| = |L(G)| - |L_i(G) \cap L_j(G)|$.*

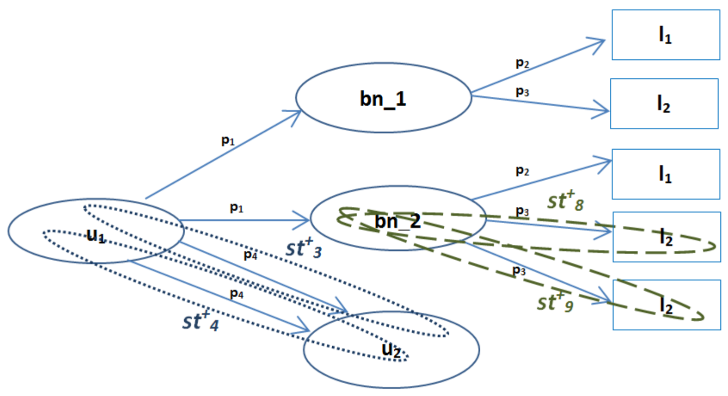


Figure 4.3: RDF graph example with edge duplication and literal node duplication.

Properties of Rule 1:

Following Lemmas 1 and 2, we can produce a set of properties which characterize an input RDF graph G, and its transformed counterpart G' resulting from applying Rule 1:

(a) $ST^{+}(G') \subseteq ST^{+}(G)$, i.e., $|ST^{+}(G')| \leq |ST^{+}(G)|$ (*reducing the number of duplicate statements, which amounts to reducing both edge and node duplications*)

(b) $L(G') \subseteq L(G)$, i.e., $|L(G')| \leq |L(G)|$ (*reducing literal node duplications*)

(c) $U(G') = U(G)$, i.e., $|U(G')| = |U(G)|$ (*preserving IRI nodes and thus all the corresponding information*)

(d) $BN(G') \subseteq BN(G)$, i.e., $|BN(G')| = |BN(G)|$ (*since we are preserving here blank nodes. Please note they are analyzed by Rule 2*)

(e) $\forall st_i^{+}, st_j^{+} \in ST^{+}(G') \;/\; i \neq j \Longrightarrow st_i^{+} \neq_{st} st_j^{+}$ (*all duplicate statements, inducing the aforementioned edge and node duplications, are eliminated*)

(f) $G' \subseteq G$, i.e., $|G'| \leq |G|$ (*since G' might suffer from other kinds of duplications which are not resolved with Rule 1*)

(g) $\|G'\| = \|G\|$ (*since minimum cardinalities are naturally equivalent*)

(h) $G' =_{RDFG} G$ (*which carries directly from the above properties*)

Example 1: Consider the RDF Graph G in Figure 4.3. One can realize that G contains a pair of duplicated edges: $st_3^{+} :< u_{1u}, p4_{\perp}^{\perp}, u_{2u} >$ and $st_4^{+} :< u_{1u}, p4_{\perp}^{\perp}, u_{2u} >$, as well as a pair of duplicated nodes: $st_8^{+} :< bn_{2bn}, p3_{\perp}^{fr}, l_{2l} >$ and $st_9^{+} :< bn_{2bn}, p3_{\perp}^{fr}, l_{2l} >$. Applying Rule 1 on G produces an RDF graph G' where both edge and node duplications have been removed as shown in Figure 4.4. As a result, G' fulfills the following properties:

(a) $|ST^{+}(G)| = 9 \wedge |ST^{+}(G')| = |ST^{+}(G)| - 2 = 7 \Longrightarrow |ST^{+}(G')| < |ST^{+}(G)|$

(b) $|L(G)| = 5 \wedge |L(G')| = |L(G)| - 1 = 4 \Longrightarrow |L(G')| < |L(G)|$

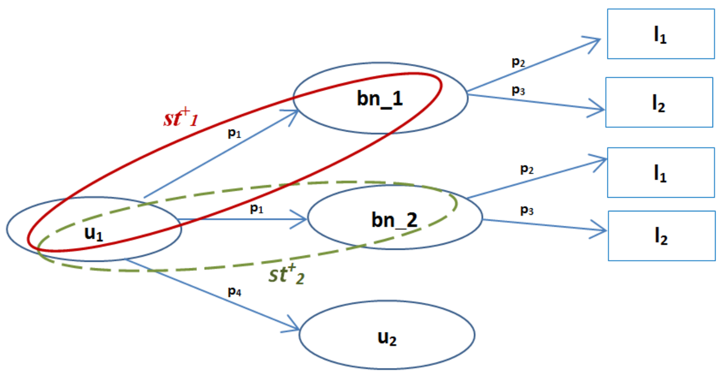


Figure 4.4: RDF graph obtained after applying R1 on the RDF Graph in Figure 4.3

(c) $|U(G)| = 2 \wedge |U(G')| = 2 \Longrightarrow |U(G')| = |U(G)|$

(d) $|BN(G)| = 2 \wedge |BN(G')| = 2 \Longrightarrow |BN(G')| = |BN(G)|$

(e) $\forall st_i^+ \in ST^+(G') \ / \ i \neq 3 \wedge i \neq 8 \Longrightarrow st_i^+ \neq_{st} st_3^+ \wedge st_i^+ \neq st_8^+$

(f) $|G'| = 7 \wedge |G| = 9$

(g) $\|G'\| = \|G\| = 4$

- **Rule 2 - Statement Containment Elimination** (**R2**): It is designed to handle extended statements and their outgoings, by eliminating edge duplications between IRIs and/or blank nodes in the outgoing statements, and eliminating node duplications where the objects of the extended statements are blank nodes linked to the outgoing statements. More formally:

  $$\forall st_i^+, st_j^+ \in ST^+(G) \ / \ i \neq j, \text{ if } st_j^+ \preceq st_i^+ \Longrightarrow remove((st_j^+ \cup O(st_j^+)), ST^+(G))$$

  Given two distinct extended RDF statements st_i^+ and st_j^+ in an RDF graph G where $st_j^+ \preceq st_i^+$, applying Rule 2 on G produces another RDF Graph G' where st_j^+ has been removed along with its outgoing statements $O(st_j^+)$♦

Lemma **3** *Given two distinct extended statements st_i^+, $st_j^+ \in ST^+(G)$ where $st_j^+ \preceq st_i^+$, applying Rule 2 on* G *produces another RDF Graph G' verifying at least one of the following features:*

- *$ST^+(G') \subseteq ST^+(G) \ / \ |ST^+(G')| = |ST^+(G)| - (1 + |O(st_j^+)|)$ (reducing the number of node duplications by $1 + |O(st_j^+)|$ where the objects of the extended statements are blank nodes).*
- *$L(G') \subseteq L(G) \ / \ |L(G')| = |L(G)| - |L(O(st_j^+))|$ (reducing the number of literal node duplications by $|L(O(st_j^+))|$ where the object of the statement is a literal).*

- *$BN(G') \subseteq BN(G)$ / $|BN(G')| = |BN(G)| - (1 + |BN(O(st_j^+))|)$ (reducing the number of blank node duplications by $1 + |BN(O(st_j^+))|$ where the statement has a blank node element).*

Proof **3** *Given two extended statements st_i^+, $st_j^+ \in ST^+(G)$ where $st_j^+ \preceq st_i^+$, applying Rule 2 on G produces an RDF graph G' which is identical to G except that in G': redundant extended statement st_j^+ and its outgoings $O(st_j^+)$ have been removed. This means that:*

- *When $st_j^+.to$ = "bn", the set of extended statements in the resulting graph G' is included in that of G, i.e., $ST^+(G') \subseteq ST^+(G)$ such that $ST^+(G') = ST^+(G) - (\{st_j^+\} \cup O(st_j^+))$ since st_j^+ and its outgoings $O(st_j^+)$ have been removed, which means $|ST^+(G')| = |ST^+(G)| - |1 + |O(st_j^+)|$.*
- *When $\forall st^+ \in O(st_j^+)$ / $st^+.to$ = "l", the set of literals in the resulting set G' is included in that of G, i.e., $L(G') \subseteq L(G)$ such that $L(G') = L(G) - (L(O(st_j^+)))$ since all the duplicated literals of outgoings $O(st_j^+)$, have been removed, which means $|L(G')| = |L(G)| - |L(O(st_j^+))|$.*
- *When ($st_j^+.to$ = "bn") $\wedge$ ($\forall st^+ \in (O(st_j^+)$ / $st^+.to$ = "bn"), the set of blank nodes in the resulting set G' is included in that of G, i.e., $BN(G') \subseteq BN(G)$ such that $BN(G') = BN(G) - (\{st_j^+.o\} \cup BN(O(st_j^+)))$ since $st_j^+.o$ and its outgoings $BN(O(st_j^+))$ have been removed, which means, $|BN(G')| = |BN(G)| - (1 + |BN(O(st_j^+))|)$.*

Lemma **4** *Given two subsets of extended statements $ST_i^+(G), ST_j^+(G) \subset ST^+(G)$ where $\forall st_i^+ \in ST_i^+(G) \wedge st_j^+ \in ST_j^+(G)$ / $st_j^+ \preceq st_i^+$, applying Rule 2 on G produces another RDF Graph G' verifying at least one of the following features:*

- *$ST^+(G') \subseteq ST^+(G)$ / $|ST^+(G')| = |ST^+(G)| - (|ST_i^+(G) \cap ST_j^+(G)| + |O(ST_i^+(G) \cap ST_j^+(G))|)$ (reducing the number of node duplications by $|ST_i^+(G) \cap ST_j^+(G)| + |O(ST_i^+(G) \cap ST_j^+(G))|$ where the objects of the extended statements are blank nodes).*
- *$L(G') \subseteq L(G)$ / $|L(G')| = |L(G)| - |L(O(ST_i^+(G) \cap ST_j^+(G)))|$ (reducing the number of literal node duplications by $|L(O(ST_i^+(G) \cap ST_j^+(G)))|$ where the objects of the extended statements of the outgoings are literals).*
- *$BN(G') \subseteq BN(G)$ / $|BN(G')| = |BN(G)| - (|BN(ST_i^+(G) \cap ST_j^+(G))| + |BN(O(ST_i^+(G) \cap ST_j^+(G)))|)$ (reducing the number of blank node duplications by $|BN_i(G) \cap BN_j(G)| + |BN(O(ST_i^+(G) \cap ST_j^+(G)))|$) where the extended statements of the outgoings have blank node elements).*

Proof **4** *Given two extended statements $ST_i^+(G), ST_j^+(G) \subset ST^+(G)$ where $\forall st_i^+ \in ST_i^+(G) \wedge st_j^+ \in ST_j^+(G) / st_j^+ \preceq st_i^+$, applying Rule 2 on G produces an RDF graph G' which is identical to G except that in G': redundant extended statements in $ST_i^+(G) \cap ST_j^+(G)$ and their outgoings $O(ST_i^+(G) \cap ST_j^+(G))$ have been removed. This means that:*

– *When* $\forall st^+ \in (ST_i^+(G) \cap ST_j^+(G))$ / $st_j^+.to =$ *"bn", the set of extended statements in the resulting graph* G' *is included in that of* G, *i.e.,* $ST^+(G') \subset ST^+(G)$ *such that* $ST^+(G') = ST^+(G) - ((ST_i^+(G) \cap ST_j^+(G)) \cup O(ST_i^+(G) \cap ST_j^+(G)))$ *since* $ST_i^+(G) \cap ST_j^+(G)$ *and its outgoings* $O(ST_i^+(G) \cap ST_j^+(G))$ *have been removed, which means* $|ST^+(G')| = |ST^+(G)| - (|ST_i^+(G) \cap ST_j^+(G)| + |O(ST_i^+(G) \cap ST_j^+(G))|)$.

– *When* $\forall st^+ \in (ST_i^+(G) \cap ST_j^+(G)) \wedge \forall st_o^+ \in O(ST_i^+(G) \cap ST_j^+(G))$ / $st^+.o = st_o^+.s \wedge st_o^+.to =$ *"l", the set of literals in the resulting set* G' *is included in that of* G, *i.e.,* $L(G') \subset L(G)$ *such that* $L(G') = L(G) - (L(O(ST_i^+(G) \cap ST_j^+(G))))$ *since all the outgoings in* $O(ST_i^+(G) \cap ST_j^+(G))$ *with duplicated literals, have been removed, which means* $|L(G')| = |L(G)| - |L(O(ST_i^+(G) \cap ST_j^+(G)))|$.

– *When* $\forall st^+ \in (ST_i^+(G) \cap ST_j^+(G)) \wedge \forall st_o^+ \in O(ST_i^+(G) \cap ST_j^+(G))$ / $st^+.o = st_o^+.s \wedge st_o^+.to =$ *"bn", the set of blank nodes in the resulting set* G' *is included in that of* G, *i.e.,* $BN(G') \subset BN(G)$ *such that* $BN(G') = BN(G) - (BN(ST_i^+(G) \cap ST_j^+(G)) \cup BN(O(ST_i^+(G) \cap ST_j^+(G))))$ *since* $BN(ST_i^+(G) \cap ST_j^+(G))$ *and its outgoings* $BN(O(ST_i^+(G) \cap ST_j^+(G)))$ *have been removed, which means* $|BN(G')| = |BN(G)| - (|BN_i(G) \cap BN_j(G)| + |BN(O(ST_i^+(G) \cap ST_j^+(G)))|)$.

Properties of Rule 2:

Following Lemmas 3 and 4, we can also produce a set of properties which characterizes an input RDF graph G and its transformed counterpart G' resulting from applying Rule 2:

(a) $ST^+(G') \subseteq ST^+(G)$, i.e., $|ST^+(G')| \leq |ST^+(G)|$ (*reducing the number of duplicate statements, which amounts to reducing both edge and node duplications*)

(b) $L(G') \subseteq L(G)$, i.e., $|L(G')| \leq |L(G)|$ (*reducing literal node duplications*)

(c) $U(G') = U(G)$, i.e., $|U(G')| = |U(G)|$ (*preserving IRI nodes and thus all the necessary information*)

(d) $BN(G') \subseteq BN(G)$, i.e., $|BN(G')| \leq |BN(G)|$ (*reducing blank node duplications*)

(e) $\forall st_i^+, st_j^+ \in ST^+(G') \implies st_j^+ \npreceq st_i^+$ (*all extended statements contained in others, inducing the aforementioned edge and node duplications, are eliminated*)

(f) $G' \subseteq G$, i.e., $|G'| \leq |G|$ (*since G' might suffer from other kinds of duplications which are not resolved with Rule 2*)

(g) $\|G'\| = \|G\|$ (*since minimum cardinalities are naturally equivalent*)

(h) $G' =_{RDFG} G$ (*which carries directly from the above properties*)

Example 2: Consider the RDF Graph G in Figure 4.4. One can realize that G contains an extended statement contained in another: $st_2^+ \preceq st_1^+$. These two extended statements induce the following node duplications in their outgoing statements:

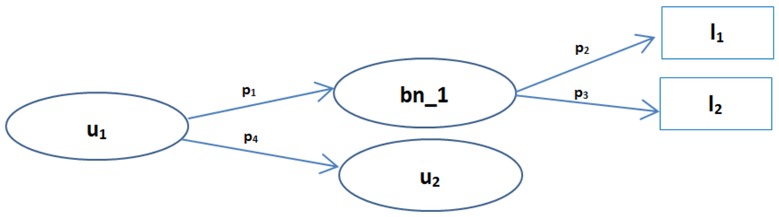


Figure 4.5: RDF graph obtained after applying R2 on the RDF graph in Figure 4.4.

$st_4^+ :< bn_{1bn}, p2_{string}^{\perp}, l_{1l} >$, $st_6^+ :< bn_{2bn}, p2_{string}^{\perp}, l_{1l} >$ and
$st_5^+ :< bn_{1bn}, p3_{\perp}^{fr}, l_{2l} >$, $st_7^+ :< bn_{2bn}, p3_{\perp}^{fr}, l_{2l} >$

Consequently, applying Rule 2 on G produces an RDF graph G' where node duplications have been removed as shown in Figure 4.5. As a result, G' fulfills the following properties:

(a) $|ST^+(G)| = 7 \wedge |O(st_2^+)| = 2$ and $|ST^+(G')| = |ST^+(G)| - 1 - |O(st_2^+)| = 4 \Longrightarrow |ST^+(G')| < |ST^+(G)|$

(b) $|L(G)| = 4 \wedge |L(O(st_2^+))| = 2$ and $|L(G')| = |L(G)| - |O(st_2^+)| = 2 \Longrightarrow |L(G')| < |L(G)|$

(c) $|U(G)| = 2 \wedge |U(G')| = 2 \Longrightarrow |U(G')| = |U(G)|$

(d) $|BN(G)| = 2 \wedge |BN(G')| = |BN(G)| - 1 \Longrightarrow |BN(G')| < |BN(G)|$

(e) $\forall st_i^+ \in ST^+(G') \ / \ i \neq j \Longrightarrow st_1^+ \not\simeq st_i^+$

(f) $|G'| = |G| = 4$

(g) $\|G'\| = \|G\| = 4$

Theorem **1** *Given an RDF graph G, applying Rules 1 and 2 on the set of extended statements of G, $ST^+(G)$, produces a graph $\tilde{G}$ which is a normalized version of G, i.e., $\tilde{G} = Norm(G)$, cf. Definition 4.4, where all logical duplications (i.e., Problems 1-3, cf. Section 3.1) have been eliminated in $\tilde{G}$.*

In goes without saying that Lemma 1 highlights the combined properties of Lemmas 1-4, which comes down to the (more general) properties of Lemmas 3-4, characterizing the relationship between an RDF graph G and its normalized counterpart $\tilde{G}$.

4.3.2 Solving Physical Disparities

Physical disparities related to namespace duplication, unused namespaces, and node order variation (presented in Section 3.2) can be eliminated from an RDF file F by applying the following transformation rules:

- **Rule 3 - Namespace Duplication Elimination** (**R3**): It is designed to eliminate namespace duplications along with corresponding namespace prefixes. More formally:

  $\forall qn_i, qn_j \in QN(F)$ / $i \neq j$, if $qn_i.ns_i = qn_j.ns_j \Longrightarrow remove(\{qn_j\}, QN(F)) \wedge$ $replace(qn_j.px_j, qn_i.px_i, ST^+(F))$♦

Lemma **5** *Given two Qnames $qn_i, qn_j \in QN(\mathrm{F})$ where $qn_i.ns_i = qn_j.ns_j$, applying Rule 3 on* F *produces an RDF file F' verifying the following features:*

- $NS(F') \subseteq NS(F)$ / $|NS(F')| = |NS(F)| - 1$ *(reducing the number of namespace duplications by 1).*
- $Px(F') \subseteq Px(F)$ / $|Px(F')| = |Px(F)| - 1$ *(reducing the number of prefixes - corresponding to the duplicated namespaces - by 1).*

Proof **5** *Given two Qnames $qn_i, qn_j \in QN(\mathrm{F})$ where $qn_i.ns_i = qn_j.ns_j$ where $qn_i = qn_j$, applying Rule 3 on F produces an RDF file F' which is identical to F except that in F': redundant qname qn_j has been removed. This means that:*

– *The set of namespaces in the resulting file F, $NS(F') \subseteq NS(F)$ / $NS(F') = NS(F) - \{qn_j.ns_j\}$ since exactly one namespace $qn_j.ns_j$ has been removed, which means $|NS(F')| = |NS(F)| - 1$.*

– *The set of prefixes in the resulting file F', $Px(F') \subseteq Px(F)$ / $Px(F') = Px(F) - \{qn_j.px_j\}$ since exactly one prefix $qn_j.px_j$ has been removed, which means $|Px(F')| = |Px(F)| - 1$.*

Lemma **6** *Given two subsets $QN_i(F), QN_j(F) \subset QN^+(F)$ where $\forall qn_i.ns_i \in NS_i(F) \wedge qn_j.ns_j \in NS_j(F)$ / $qn_i.ns_i = qn_j.ns_j$, applying Rule 3 on F produces an RDF file F' verifying the following features:*

- $NS(F') \subseteq NS(F)$ / $|NS(F')| = |NS(F)| - |NS_i(F) \cap NS_j(F)|$ *(reducing the number of namespace duplications by $|NS_i(F) \cap NS_j(F)|$).*
- $Px(F') \subseteq Px(F)$ / $|Px(F')| = |Px(F)| - |Px_i(F) \cap Px_j(F)|$ *(reducing the number of prefixes - corresponding to the duplicated namespaces - by $|Px_i(F) \cap Px_j(F)|$).*

Proof **6** *Given two subsets of Qnames $QN_i(F), QN_j(F) \subset QN^+(F)$ where $\forall qn_i.ns_i \in NS_i(F) \wedge qn_j.ns_j \in NS_j(F)$ / $qn_i.ns_i = qn_j.ns_j$, applying Rule 3 on F produces an RDF file F' which is identical to F except that in F': redundant qnames $QN_i(F) \cap QN_j(F)$ have been removed. This means that:*

- *The set of namespaces in the resulting file* F, $NS(F') \subseteq NS(F)$ / $NS(F') = NS(F) - (NS_i(F) \cap NS_j(F))$ *since namespaces in* $NS_i(F) \cap NS_j(F)$ *have been removed, which means* $|NS(F')| = |NS(F)| - |NS_i(F) \cap NS_j(F)|$.
- *The set of prefixes in the resulting file* F', $Px(F') \subseteq Px(F)$ / $Px(F') = Px(F) - (Px_i(F) \cap Px_j(F))$ *since prefixes in* $Px_i(F) \cap Px_j(F)$ *have been removed, which means* $|Px(F')| = |Px(F)| - |Px_i(F) \cap Px_j(F)|$.

Properties of Rule 3:

Following Lemmas 5 and 6, we can produce a set of properties which characterizes an input RDF file F and its transformed counterpart F' resulting from applying Rule 3:

(a) $\forall ns_i, ns_j \in NS(F')$ / $i \neq j \Longrightarrow ns_i \neq ns_j$ (*eliminating all namespace duplications*)

(b) $NS(F') \subseteq NS(F)$, i.e., $|NS(F')| \leq |NS(F)|$ (*reducing the number of namespaces, as a result of eliminating namespace duplications*)

(c) $\forall px_i, px_j \in PX(F')$ / $i \neq j \Longrightarrow px_i \neq px_j$ (*eliminating all prefixes corresponding to the duplicate namespaces*)

(d) $Px(F') \subseteq Px(F)$, i.e., $Px(F') \subseteq Px(F)$, i.e.,$|Px(F')| \leq |Px(F)|$ (*reducing the number of prefixes, as a result of eliminating prefix duplications*)

(e) Corresponding RDF graphs remain the same: $G' = G$, i.e., $|G'| = |G|$ (*since extended statements are not affected at the logical level*)

(f) $F' =_{RDFF} F$ (*naturally carries from the above properties*)

Example 3: Consider the RDF file F in Figure 3.3. One can realize that F contains the following Qnames with namespace duplications and corresponding prefixes:

$qn_1 < px_1, ns_1 > \Longrightarrow px_1 =$ "*ex*", $ns_1 =$ "*http : //example.org/stuff/*1.0/" (line 4) $\wedge$

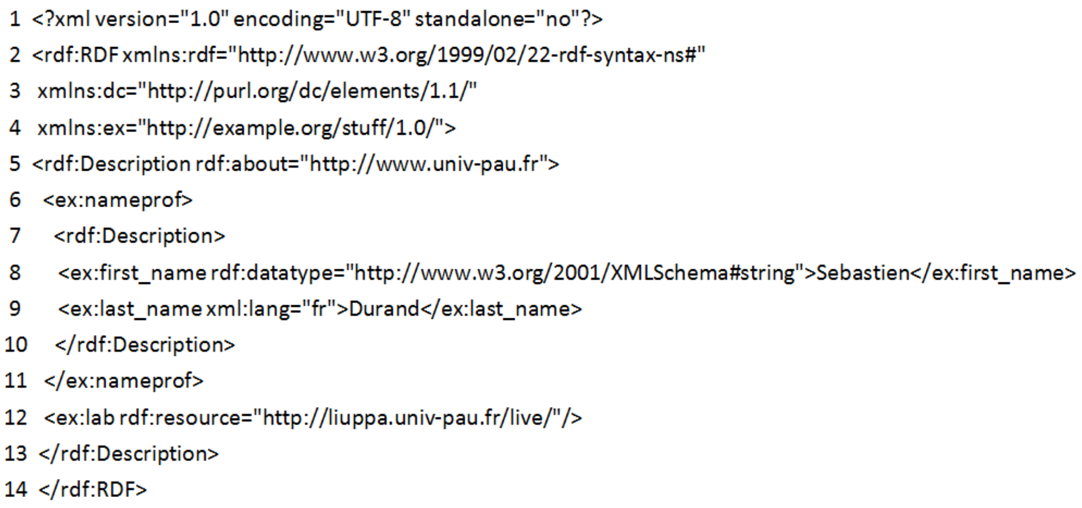
$qn_2 < px_2, ns_2 > \Longrightarrow px_2 =$ "*ex*1", $ns_2 =$ "*http : //example.org/stuff/*1.0/" (line 5)

Applying Rule 3 on F produces an RDF file F' where namespace duplications with corresponding prefixes have been removed as shown in Figure 4.6. As a result, F' fulfills the following properties:

(a) $|NS(F)| = 4 \wedge |NS(F')| = |NS(F)| - 1 = 3 \Longrightarrow |NS(F')| < |NS(F)|$
(considering the default "http://www.w3.org/1999/02/22-rdf-syntax-ns" as a name space)

(b) $\forall qn_i.ns_i \in NS(F')$ / $i \neq 1 \Longrightarrow qn_1.ns_1 \neq qn_i.ns_i$

(c) $|Px(F)| = 4 \wedge |Px(F')| = |Px(F)| - 1 = 3 \Longrightarrow |Px(F')| < |Px(F)|$
(considering the default "rdf" as prefix)



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:ex="http://example.org/stuff/1.0/">
<rdf:Description rdf:about="http://www.univ-pau.fr">
 <ex:nameprof>
  <rdf:Description>
   <ex:first_name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Sebastien</ex:first_name>
   <ex:last_name xml:lang="fr">Durand</ex:last_name>
  </rdf:Description>
 </ex:nameprof>
 <ex:lab rdf:resource="http://liuppa.univ-pau.fr/live/"/>
</rdf:Description>
</rdf:RDF>
```

Figure 4.6: RDF file obtained after applying R1, R2 and R3 on the RDF file in Figure 3.3

(d) $\forall px_i \in Px(F') \ / \ i \neq 1 \Longrightarrow px_1 \neq px_i$

(e) $|G'| = |G| = 4$ (since the files' RDF graphs remain unchanged)

- **Rule 4 - Unused Namespace Elimination (R4)**: It is designed to eliminate unused namespaces[1] with their respective prefixes. More formally:

  $\forall qn_i.ns_i \in NS(F)$, if $qn_i.ns_i \notin NS(G) \Longrightarrow remove(\{qn_i\}, QN(F))$.

  Given qn_i in an RDF file F where $qn_i.ns_i$ is not used in any RDF statement in F, applying R4 on F produces another RDF file F' where unused namespace $qn_i.ns_i$ and its respective prefix px_i have been removed⧫

Lemma 7 *Given a Qname $qn_i \in QN(F)$ where $qn_i.ns_i \notin NS(G)$, applying Rule 4 on F produces an RDF file F' verifying the following feature:*

$QN(F') \subseteq QN(F) \ / \ |QN(F')| = |QN(F)| - 1$ (reducing the number of unused QNames by 1).

Proof 7 *Given a Qname $qn_i \in QN(F)$ where $qn_i.ns_i \notin NS(G)$, applying Rule 4 on F produces an RDF file F' which is identical to F except that in F': unused qname qn_i has been removed. This means that the set of qnames in the resulting file F, $QN(F') \subseteq QN(F) \ / \ QN(F') = QN(F) - \{qn_i\}$ since exactly one qname qn_i has been removed, which means $|QN(F')| = |QN(F)| - 1$.*

[1]An unused namespace is a namespace which is mention in the serialization file but which is not use in any of the statements, it is means, they will not appear in the Graph.

LEMMA **8** *Given a subset $QN_i(F) \subset QN(F)$ where $\forall qn_i.ns_i \in QN_i(F)$ / $qn_i.ns_i \notin NS(G)$, i.e., $UNS(F) = UNS(F) \cup \{qn_i.ns_i\}$, applying Rule 4 on F produces an RDF file F' verifying the following feature:*

$QN(F') \subseteq QN(F)$ / $|QN(F')| = |QN(F)| - |QN(UNS(F))|$ *(reducing the number of unused QNames by $|QN(UNS(F))|$).*

PROOF **8** *Given a set Qname $QN_i(F) \subset QN(F)$ where $\forall qn_i.ns_i \in QN_i(F)/qn_i.ns_i \notin NS(G)$, i.e., $UNS(F) = UNS(F) \cup \{qn_i.ns_i\}$, applying Rule 4 on F produces an RDF file F' which is identical to F except that in F': unused qnames set $QN(UNS(F))$ has been removed. This means that the set of qnames in the resulting file F, $QN(F') \subseteq QN(F)$ / $QN(F') = QN(F) - (QN(UNS(F)))$ since unused qnames in $QN(UNS(F))$ have been removed, which means $|QN(F')| = |QN(F)| - |QN(UNS(F))|$.*

Properties of Rule 4:

Following Lemmas 7 and 8, we can produce a set of properties which characterizes an input RDF file F and its transformed counterpart F' resulting from applying Rule 4:

(a) $NS(F') \subseteq NS(F)$, i.e., $|NS(F')| \leq |NS(F)|$ (*reducing the number of unused namespaces*)

(b) $Px(F') \subseteq Px(F)$, i.e.,$|Px(F')| \leq |Px(F)|$ (*reducing the number of unused prefixes*)

(c) $NS(G') \subseteq NS(F')$, i.e., $|NS(G')| = |NS(F')|$ (*the number of used namespaces in the RDF Graph becomes equal to that in the corresponding RDF file*)

(d) Corresponding RDF graphs remain the same: $|G'| = |G|$ (*since extended statements are not affected at the logical level*)

(e) $F' =_{RDFF} F$ (*naturally carries from the above properties*)

Example 4: Consider the RDF file F in Figure 4.6. One can realize that F contains the following unused namespace and corresponding prefix:

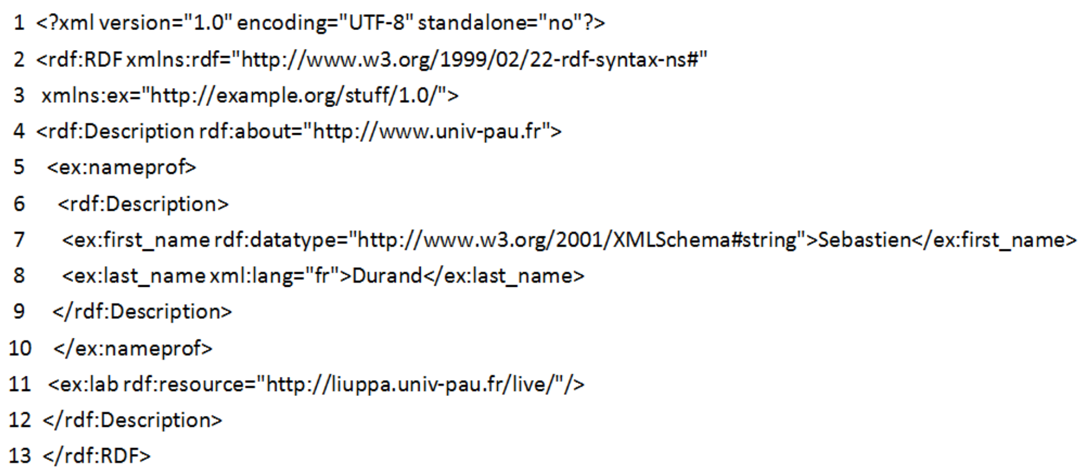
$ns_3 =$ "$http://purl.org/dc/elements/1.1/$" $\rightarrow px_3 =$ "dc" (line 3)

Applying Rule 4 on F produces an RDF file F' where the unused namespace and its corresponding prefix have been removed as shown in Figure 4.7. As a result, F' fulfills the following properties:

(a) $|NS(F)| = 3 \wedge |NS(F')| = |NS(F)| - 1 = 2 \Longrightarrow |NS(F')| < |NS(F)|$

(b) $|Px(F)| = 3 \wedge |Px(F')| = |Px(F)| - 1 = 2 \Longrightarrow |Px(F')| < |Px(F)|$

(c) $|NS(F)| = 2 \wedge |NS(G')| = 2 \Longrightarrow |NS(F')| = |NS(G')|$



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:ex="http://example.org/stuff/1.0/">
<rdf:Description rdf:about="http://www.univ-pau.fr">
  <ex:nameprof>
    <rdf:Description>
     <ex:first_name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Sebastien</ex:first_name>
     <ex:last_name xml:lang="fr">Durand</ex:last_name>
   </rdf:Description>
  </ex:nameprof>
 <ex:lab rdf:resource="http://liuppa.univ-pau.fr/live/"/>
</rdf:Description>
</rdf:RDF>
```

Figure 4.7: RDF file obtained after applying R4 on the RDF file in Figure 4.6.

(d) $|G'| = |G| = 4$ (since the files' RDF graphs remain unchanged)

- **Rule 5 - Reordering (R5)**: It is designed to solve the varying node order problem by imposing a predefined (user-chosen) order on all statements of an RDF file F'. More formally:

  $$\forall st_i^+, st_j^+ \in ST^+(F), order(st_i^+, st_j^+, F, \tilde{p}) \Longrightarrow st_i^+ <_{\tilde{p}} st_j^+$$

  Given the extended statements in an RDF file F, where stated in F following an initial order, applying Rule 5 using the order function (cf. Section 4.1.2) with ordering parameter $\tilde{p}$ (based on our *statement expression order* detailed in Section 4.5.2.2) on the two extended statements in F produces an RDF file F' which is equal to F, $F' =_{RDFF} F$, where all the statements have been ordered following the (user-chosen) order type parameter $\tilde{p}$♦

The parameter $\tilde{p}$ is a tuple composed of ***indexing order*** "*iorder*" and ***sorting criteria*** "*sortc*" (the values of these two elements are detailed in Section 4.5.2.2), represented by $\tilde{p} :< iorder, sortc >$. The default value for the parameter $\tilde{p}$ in our proposal is $< sop, asc >$ representing an ascending order of statements w.r.t. their subjects / objects / predicates (sop).

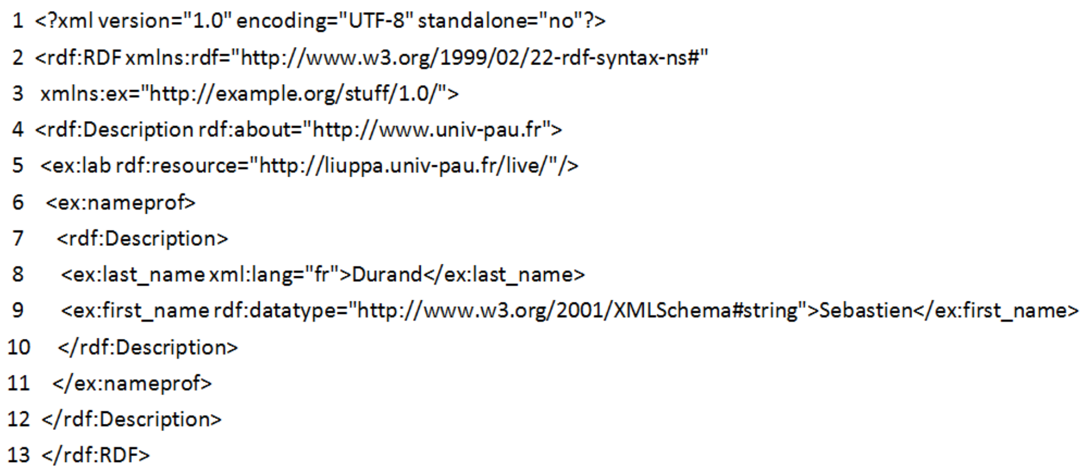
Properties of Rule 5:

(a) $\tilde{F} =_{RDFF} F$ (*both files having equal RDF graphs and the same encoding format*)

(b) $|F| = |\tilde{F}|$ (*both files having the same number of statements*)

(c) The only difference between F and $\tilde{F}$ is in statement ordering, noted $order(ST^{+}(F)) \neq_{\tilde{p}} order(ST^{+}(\tilde{F}))$ (*they are different according to their respective $\tilde{p}$ parameter order*)

Example 5: Consider RDF file F in Figure 4.7, ordering $ST^{+}(F)$ using our default ordering parameter $\tilde{p} :< SOP, asc >$ (Subject-Object-Predicate in ascending order following our sorting process detailed in Section 4.5.2.2) produces an RDF file F' where all statements have been re-ordered accordingly, as shown in Figure 4.8.



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:ex="http://example.org/stuff/1.0/">
<rdf:Description rdf:about="http://www.univ-pau.fr">
 <ex:lab rdf:resource="http://liuppa.univ-pau.fr/live/"/>
  <ex:nameprof>
   <rdf:Description>
    <ex:last_name xml:lang="fr">Durand</ex:last_name>
    <ex:first_name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Sebastien</ex:first_name>
   </rdf:Description>
  </ex:nameprof>
</rdf:Description>
</rdf:RDF>
```

Figure 4.8: RDF file result after applying R5 in Figure 4.7.

Theorem **2** *Given an RDF file F, applying Rules 3, 4 and 5 on F produces a file $\tilde{F}$ which is a normalized version of F, i.e., $\tilde{F}$ = Norm(F), cf. Definition 4.4, where all physical disparities (i.e., Problems 4-6, cf. Section 3.2) have been eliminated in $\tilde{F}$.*

In goes without saying that Lemma 2 highlights the combined properties of Lemmas 5-8, characterizing the relationship between an RDF file F and its normalized counterpart $\tilde{F}$.

Table 4.7 provides a snapshot of all normalization rules with their properties. Note that the problems related to element types and language tags can also be related to the semantic meaning of corresponding elements, and will be further investigated using dedicated semantic-aware transformation rules which we report to a subsequent study.

Table 4.6: RDF description normalization rules

#	Notation	Features of the Lemma	Properties
Logical Rules			
R1	$\forall st_i^+, st_j^+ \in ST^+(G)$ / $i \neq j$, if $st_i^+ =_{st} st_j^+ \Longrightarrow remove(\{st_j^+\}, ST^+(G))$	Lemma 1 - $ST^+(G') \subseteq ST^+(G)$ / $\|ST^+(G')\| = \|ST^+(G)\| - 1$ - $L(G') \subseteq L(G)$ / $\|L(G')\| = \|L(G)\| - 1$ Lemma 2 - $ST^+(G') \subseteq ST^+(G)$ / $\|ST^+(G')\| = \|ST^+(G)\| - \|ST_i^+(G) \cap ST_j^+(G)\|$ - $L(G') \subseteq L(G)$ / $\|L(G')\| = \|L(G)\| - \|L_i(G) \cap L_j(G)\|$	(a) $ST^+(G') \subseteq ST^+(G)$, i.e., $\|ST^+(G')\| \leq \|ST^+(G)\|$ (b) $L(G') \subseteq L(G)$, i.e., $\|L(G')\| \leq \|L(G)\|$ (c) $U(G') = U(G)$, i.e., $\|U(G')\| = \|U(G)\|$ (d) $BN(G') \subseteq BN(G)$, i.e., $\|BN(G')\| = \|BN(G)\|$ (e) $\forall st_i^+, st_j^+ \in ST^+(G')$ / $i \neq j \Longrightarrow st_i^+ \neq_{st} st_j^+$ (f) $G' \subseteq G$, i.e., $\|G'\| \leq \|G\|$ (g) $\\|G'\\| = \\|G\\|$ (f) $\tilde{G} =_{RDFG} G$
R2	$\forall st_i^+, st_j^+ \in ST^+(G)$ / $i \neq j$, if $st_j^+ \preceq st_i^+ \Longrightarrow remove((st_j^+ \cup O(st_j^+)), ST^+(G))$	Lemma 3 - $ST^+(G') \subseteq ST^+(G)$ / $\|ST^+(G')\| = \|ST^+(G)\| - (1 + \|O(st_j^+)\|)$ - $L(G') \subseteq L(G)$ / $\|L(G')\| = \|L(G)\| - \|L(O(st_j^+))\|$ - $BN(G') \subseteq BN(G)$ / $\|BN(G')\| = \|BN(G)\| - (1 + \|BN(O(st_j^+))\|)$ Lemma 4 - $ST^+(G') \subseteq ST^+(G)$ / $\|ST^+(G')\| = \|ST^+(G)\| - (\|ST_i^+(G) \cap ST_j^+(G)\| + \|O(ST_i^+(G) \cap ST_j^+(G))\|)$ - $L(G') \subseteq L(G)$ / $\|L(G')\| = \|L(G)\| - \|L(O(ST_i^+(G) \cap ST_j^+(G)))\|$ - $BN(G') \subseteq BN(G)$ / $\|BN(G')\| = \|BN(G)\| - (\|BN(ST_i^+(G) \cap ST_j^+(G))\| + \|BN(O(ST_i^+(G) \cap ST_j^+(G)))\|)$	(a) $ST^+(G') \subseteq ST^+(G)$, i.e., $\|ST^+(G')\| \leq \|ST^+(G)\|$ (b) $L(G') \subseteq L(G)$, i.e., $\|L(G')\| \leq \|L(G)\|$ (c) $U(G') = U(G)$, i.e., $\|U(G')\| = \|U(G)\|$ (d) $BN(G') \subseteq BN(G)$, i.e., $\|BN(G')\| \leq \|BN(G)\|$ (e) $\forall st_i^+, st_j^+ \in ST^+(G') \Longrightarrow st_j^+ \npreceq st_i^+$ (f) $G' \subseteq G$, i.e., $\|G'\| \leq \|G\|$ (g) $\\|G'\\| = \\|G\\|$ (h) $\tilde{G} =_{RDFG} G$

Table 4.7: RDF description normalization rules

#	Notation	Features of the Lemma	Properties
		Physical Rules	
R3	$\forall qn_i, qn_j \in QN(F)$ / $i \neq j$, if $qn_i.ns_i = qn_j.ns_j \Longrightarrow$ $remove(\{qn_j\}, QN(F)) \wedge replace$ $(qn_j.px_j,\ qn_i.px_i,\ F)$	Lemma 5 - $NS(F') \subseteq NS(F)$ / $\|NS(F')\| = \|NS(F)\| - 1$ - $Px(F') \subseteq Px(F)$ / $\|Px(F')\| = \|Px(F)\| - 1$ Lemma 6 - $NS(F') \subseteq NS(F)$ / $\|NS(F')\| = \|NS(F)\| - \|NS_i(F) \cap NS_j(F)\|$ - $Px(F') \subseteq Px(F)$ / $\|Px(F')\| = \|Px(F)\| - \|Px_i(F) \cap Px_j(F)\|$	(a) $\forall ns_i, ns_j \in NS(\tilde{F})$ / $i \neq j \Longrightarrow ns_i \neq ns_j$ (b) $NS(F') \subseteq NS(F)$, i.e., $\|NS(F')\| \leq \|NS(F)\|$ (c) $\forall px_i, px_j \in PX(F')$ / $i \neq j \Longrightarrow px_i \neq px_j$ (d) $Px(F') \subseteq Px(F)$, i.e., $\|Px(F')\| \leq \|Px(F)\|$ (e) $\|G'\| = \|G\|$ (f) $\tilde{F} =_{RDFF} F$
R4	$\forall qn_i.ns_i \in NS(F)$ if $qn_i.ns_i \notin NS(G) \Longrightarrow$ $remove(\{qn_i\}, QN(F))$	Lemma 7 - $QN(F') \subseteq QN(F)$ / $\|QN(F')\| = \|QN(F)\| - 1$ Lemma 8 - $QN(F') \subseteq QN(F)$ / $\|QN(F')\| = \|QN(F)\| - \|QN(UNS(F))\|$	(a) $NS(F') \subseteq NS(F)$, i.e., $\|NS(F')\| \leq \|NS(F)\|$ (b) $Px(F') \subseteq Px(F)$, i.e., $\|Px(F')\| \leq \|Px(F)\|$ (c) $NS(G') \subseteq NS(F')$, i.e., $\|NS(G')\| = \|NS(F')\|$ (d) $\|G'\| = \|G\|$ (e) $\tilde{F} =_{RDFF} F$
R5	$\forall st_i^+, st_j^+ \in ST^+(F)$, $order(st_i^+, st_j^+,\ F, \tilde{p}) \Longrightarrow$ $st_i^+ <_{\tilde{p}} st_j^+$		(a) $\tilde{F} =_{RDFF} F$ (b) $\|F\| = \|\tilde{F}\|$ (c) $ord(ST^+(F)) \neq_{\tilde{p}} ord(ST^+(\tilde{F}))$

4.4 Normalization Properties

Based on the individual normalization rules' properties (highlighted based on their corresponding lemmas in the previous section) allowing both logical and physical normalization, we develop and discuss in this section the general properties characterizing the quality of our integrated normalization approach.

Definition **4.5 (Property 1: *Completeness*)** *An RDF description D and its transformed counterpart D', D' is said to be complete regarding D if D' preserves and does not lose any information w.r.t. D, i.e., each resource, statement, and namespace of D has a corresponding resource, statement, and namespace in D'. More formally:*

$$D' \triangleq D \Longleftrightarrow \begin{cases} U(D') = U(D) \\ BN(D') \subseteq BN(D) \\ L(D') \subseteq L(D) \\ ST^{+}(D') \subseteq ST^{+}(D) \\ NS(D') \subseteq NS(D) \\ \|D'\| = \|D\| \end{cases}$$

Lemma ***9*** . *Given an RDF description D, its normalized counterpart $\tilde{D}$ is complete w.r.t. D.*

Proof ***9*** . *Given an RDF description D and its normalized counterpard $\tilde{D}$, the following properties are satisfied:*

- *Properties resulting from applying rules R1 and R2:*
  - *The sets of IRIs in the original and resulting RDF descriptions D and $\tilde{D}$ are equal, i.e., $U(\tilde{D}) = U(D)$ / $|U(\tilde{D})| = |U(D)|$ since $\forall u \in U(\tilde{D}) \Rightarrow u \in U(D)$.*
  - *The set of blank nodes in the resulting RDF description $\tilde{D}$ is included in that of D, i.e., $BN(\tilde{D}) \subseteq BN(D)$ / $|BN(\tilde{D})| \leq |BN(D)|$ since $\forall bn \in BN(\tilde{D}) \Rightarrow bn \in BN(D)$.*
  - *The set of literals in the resulting RDF description $\tilde{D}$ is included in that of D, i.e., $L(\tilde{D}) \subseteq L(D)$ / $|L(\tilde{D})| \leq |L(D)|$ since $\forall l \in L(\tilde{D}) \Rightarrow l \in L(D)$.*
  - *The set of extended statements in the resulting RDF description $\tilde{D}$ is included in that of D, i.e., $ST^{+}(\tilde{D}) \subseteq ST^{+}(D)$ / $|ST^{+}(\tilde{D})| \leq |ST^{+}(D)|$ since $(\forall st^{+} \in ST^{+}(\tilde{D}) \Rightarrow st^{+} \in ST^{+}(D)) \vee (\forall st_i^{+}, st_j^{+} \in ST^{+}(\tilde{D}), st_i^{+} \preceq st_j^{+} \Rightarrow st_i^{+} \in ST^{+}(D))$.*
- *Properties resulting from applying rules R3 and R4:*

   – *The set of namespaces in the resulting RDF description $\tilde{D}$ is included in that of D, i.e., $NS(\tilde{D}) \subseteq NS(D)$ / $|NS(\tilde{D})| \leq |NS(D)|$ since $\forall ns \in NS(\tilde{D}) \Rightarrow ns \in NS(D)$.*
   – *The set of namespaces in the resulting RDF description $\tilde{D}$ is equal to that of D, i.e., $NS(\tilde{F}) = NS(G)$ / $|NS(\tilde{F})| = |NS(G)|$ since $\tilde{F}$ is one serialization of D and $\tilde{G}$ is the Graph of D.*

- *The minimum cardinality of the resulting RDF description $\tilde{D}$ has to be the same of the minimum cardinality of D since $\tilde{D}$ is the same RDF description without duplications and unused information.*

Therefore, we conclude that $\tilde{D}$ is complete w.r.t. D ⧫

Definition **4.6 (Property 2: *Minimality*)** *An RDF Description D is said to be minimal, noted by* Dmin*, if all the resources, statements, and namespaces of D are unique (i.e., they do not have duplicates in D) and all the namespaces are used (i.e., there are no unused namespaces). More formally:*

$$\text{Dmin} \Longleftrightarrow \forall i \neq j \begin{cases} \forall u_i, u_j \in U(D) \Longrightarrow u_i \neq u_j \\ \forall bn_i, bn_j \in BN(D) \Longrightarrow bn_i \neq bn_j \\ \forall st_i^+, st_j^+ \in ST^+(D) \Longrightarrow st_i^+ \neq_{st} st_j^+ \\ \forall ns_i, ns_j \in NS(D) \Longrightarrow ns_i \neq ns_j \Longrightarrow UNS(D) = \varnothing \end{cases}$$

Lemma ***10*** . *Given an RDF description D, its normalized counterpart $\tilde{D}$ is minimal.*

Proof ***10*** . *Given an RDF description D, applying rules Rules 1, 2 and 3 produces a normalized RDF description $\tilde{D}$ verifying the following properties:*

* $\forall u_i, u_j \in U(\tilde{D}) \Longrightarrow u_i \neq u_j$. *Following Rules 1 and 2 (properties "c" and "e")*

* $\forall bn_i, bn_j \in BN(\tilde{D}) \Longrightarrow bn_i \neq bn_j$. *Following Rules 1 and 2 (properties "d" and "e")*

* $\forall st_i^+, st_j^+ \in ST^+(\tilde{D}) \Longrightarrow st_j^+ \neq_{st} st_i^+$. *Following Rules 1 and 2 (properties "a" and "e")*

* $\forall ns_i, ns_j \in NS(\tilde{D}) \Longrightarrow ns_i \neq ns_j$. *Following Rule 3 (properties "a" and "b")*

Therefore, we conclude that $\tilde{D}$ is minimal ⧫

Definition **4.7 (Property 3: *Compliance*)** *An RDF file F is said to be compliant with the RDF standard if: i) its corresponding RDF graph G is valid w.r.t. the RDF standard, i.e., G's structure remains compliant with RDF serialization standards (e.g., RDF/XML)), ii) all* *extended statements in G also appear in F and, iii) all namespaces used in G also appear in F. More formally:*

$$F \rhd RDF \Longleftrightarrow \begin{cases} G \rhd RDF \\ |ST^{+}(G)| = |ST^{+}(F)| \\ |NS(G)| = |NS(F)| \end{cases}$$

Lemma ***11*** . *Given an RDF file F, its normalized counterpart $\tilde{F}_i$ is compliant with the RDF standard.*

Proof ***11*** . *Given and RDF file F and its normalized counterpart $\tilde{F}$, with G and $\tilde{G}$ representing their corresponding RDF graphs:*

- $\forall st_i^{+}, st_j^{+} \in ST^{+}(G)\ /\ i \neq j$, *if* $st_i^{+} =_{st} st_j^{+} \Longrightarrow \tilde{G}$ *will be identical to G except that st_j^{+} has been removed from $\tilde{G}$ (satisfying R1). Hence, if* $G \rhd RDF \Longrightarrow \tilde{G} \rhd RDF$
- $\forall st_i^{+}, st_j^{+} \in ST^{+}(G)\ /\ i \neq j$, *if* $st_j^{+} \preceq st_i^{+} \Longrightarrow \tilde{G}$ *will be identical to G except that* $st_j^{+} \cup O(st_j^{+})$ *have been removed from $\tilde{G}$ (satisfying R2). Hence, if* $G \rhd RDF \Longrightarrow \tilde{G} \rhd RDF$
- $\forall qn_i, qn_j \in QN(F)\ /\ i \neq j$, *if* $qn_i.ns_i = qn_j.ns_j \Longrightarrow \tilde{F}$ *will be identical to F except that qn_j has been removed from $\tilde{F}$ (satisfying R3). Hence, if* $F \rhd RDF \Longrightarrow \tilde{F} \rhd RDF$
- $\forall qn_i.ns_i \in NS(F)$ *if* $qn_i.ns_i \notin NS(G) \Longrightarrow \tilde{F}$ *will be identical to F except that qn_j has been removed from $\tilde{F}$ (satisfying R4). Hence, if* $(F \wedge G) \rhd RDF \Longrightarrow (\tilde{F} \wedge \tilde{G}) \rhd RDF$
- $\forall st_i^{+}, st_j^{+} \in ST^{+}(F)$, $order(st_i^{+}, st_j^{+}, F, \tilde{p}) \Longrightarrow \tilde{F}$ *will be identical to F except that in* $\tilde{F} : st_i^{+} <_{\tilde{p}} st_j^{+}$ *after ordering (satisfying R5). Hence, if* $F \rhd RDF \Longrightarrow \tilde{F} \rhd RDF$

Therefore, given the above such that $|ST^{+}(G)| = |ST^{+}(F)|$ and $|NS(G)| = |NS(F)|$, we conclude that $\tilde{F}$ is compliant with the RDF standard ♦

Definition **4.8 (Property 4: *Consistency*)** *Given an RDF description D and its transformed counterpart D', D' is said to be consistent if D' verifies all three properties: i) completeness w.r.t. D, ii) minimality, and iii) compliance w.r.t. the RDF standard; which, combined, ensure the data quality of the description. Formally:*

$$D' \text{ is consistent} \Longleftrightarrow \begin{cases} D' \triangleq D & holds \\ D' = D'min & holds \\ F' \rhd RDF & holds \end{cases}$$

Lemma ***12*** . *Given an RDF description D, its normalized counterpart $\tilde{D}$ is consistent w.r.t. D.*

PROOF ***12*** . *Given an RDF description D and its normalized counterpart $\tilde{D}$:*

- *$\tilde{D} \triangleq D$ following Lemma 9, i.e., $\tilde{D}$ is complete w.r.t. D since it preserves and does not loose any information w.r.t. D.*
- *$\tilde{D} = Dmin$ following Lemma 10, i.e., $\tilde{D}$ is minimal such that all statements, resources, and namespaces are unique in $\tilde{D}$.*
- *Given $\tilde{F}$ the serialization of $\tilde{D}$, $\tilde{F} \rhd RDF$ following Lemma 11, i.e., $\tilde{F}$ is compliant with the RDF standard.*

Therefore, we conclude that $\tilde{D}$ is consistent ♦

Verifying RDF description *consistency* means that we will be preserving all the IRIs and namespaces (with their prefixes) in the normalized RDF description which can be reused later. This corresponds to the notion of *information reusability* which is discussed in existing studies[1]. Through the shareability of the RDF standard, the resources will support the reusability of metadata on the Web. With reusability, RDF descriptions can be more robust (as discussed in the Sections 1 and 4.2), while saving on storage space by avoiding duplications.

4.5 RDF Normalization Process

The overall architecture of our R2NR (RDF to Normalized RDF) framework is depicted in Figure 4.9. It consists of two main components: i) Logical Normalization and ii) Physical Normalization. In short, both components have different algorithms to control and manage the redundancies and disparities discussed in Section 3, by implementing our normalization rules developed in Section 4.3. R2NR accepts as input: a) the RDF graph (logical representation) or RDF file (physical representation) to be normalized, and b) user parameters related to the RDF output form and prefix renaming, enabling the user to tune the results according to her/his requirements.

Note that the development of separate components is a design choice to: i) emphasize the modularity of our approach (allowing to easily integrate additional algorithms or modules in the future), and ii) enable the user to easily customize the normalization process (depending on the application at hand). In the following, we describe each component in more details.

[1]This is comparable to the notion of *map function* in [Gea11] where authors verify that RDF files have valid structures and contain necessary information (namely IRIs), as a pre-processing step before storage in an RDF database. Yet, the authors in [Gea11] focus on the general theoretical foundations of RDF processing, and do not specifically target normalization problems (cf. Section 4.2.2).

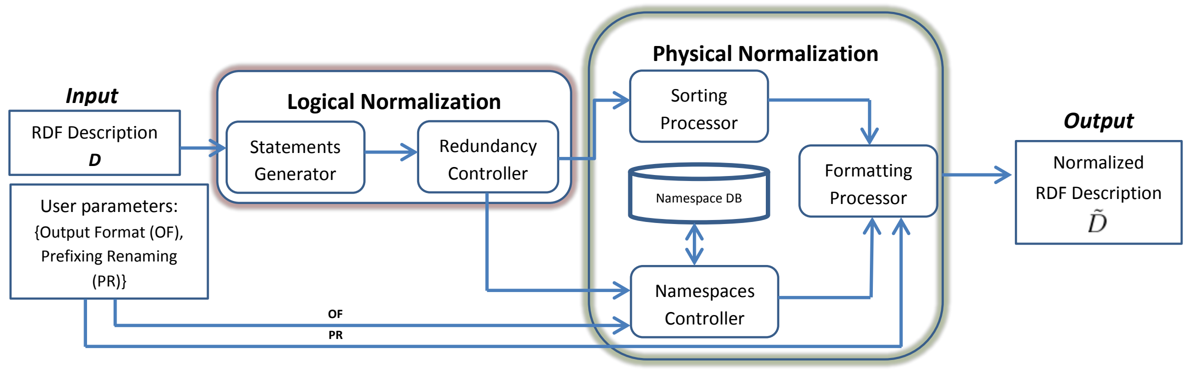


Figure 4.9: Overall architecture of our R2NR framework.

4.5.1 Logical Normalization

The first step in our normalization process is to perform logical normalization, allowing to eliminate all logical redundancies (discussed in Section 3.1) from nodes and edges of the RDF graph, and obtain extended statements without duplications (cf. Rules 1 and 2). For this purpose, we divide logical normalization in two sub-components:

4.5.1.1 Statement Generator (SG)

It implements a preprocessing step, generating the *extended statements* (see Def. 4.1) from the input RDF file.

4.5.1.2 Redundancy Controller (RC)

It implements the core logical normalization process, detecting and eliminating edge and node duplications in the RDF graph. The input of this sub-component is the list of *extended statements*. We provide the pseudo-code of the redundancy controller in Algorithm 1. The algorithm starts by detecting and erasing the redundancies in statements that contain IRIs or literals.

Consequently, it removes the statements with duplicated blank nodes (bn) (as well as all the outgoings O derived of the bn) using Operator 1 and based on normalization Rules 1 and 2.

Algorithm 1 Redundancy Controller

```
Input: $ST^+[]$ //List of Extended Statements of the RDF Description
Output: $ST^+[]$ //List of Extended Statements without duplication
 1: N=$st^+$.length(); //Number of Statements in the list
 2: for i=1, i ≤ N, i++ do
 3:     for j=i+1, j ≤ N, j++ do
 4:         if $st^+$[i].to = “IRI” or $st^+$[i].to = “literal” and $st^+[i] =_{st} st^+[j]$ then
 5:             $remove(st^+_j, ST^+[])$; // remove statement duplication - Rule 1
 6:         else
 7:             if $st^+[j]$.to = “bn” and $st^+[i]$.to = “bn” and $st^+[i]$.s = $st^+[j]$.s and $st^+[i]$.p = $st^+[j]$.p and ($st^+[i] \preceq st^+[j]$
                or $st^+[j] \preceq st^+[i]$) then
 8:                 $remove((st^+_j \cup O(st^+_j)), ST^+[])$; // remove blank node duplication - Rule 2
 9:             else
10:                 if $st^+[i]$.o = $st^+[j]$.o then
11:                     $remove(st^+_j, ST^+[])$ // remove statement duplication - Rule 1
12: return $ST^+[]$
```

4.5.2 Physical Normalization

The second step in our normalization process is to perform physical normalization by handling serialization disparities (discussed in Section 3.2, cf. Rules 3, 4, and 5). It is divided into three sub-components based on the types of physical disparities being processed:

4.5.2.1 Namespaces Controller (NC)

It controls namespace duplication by erasing redundant namespaces (Rule 3) and unused namespaces (Rule 4) in the RDF file. This component takes as input the prefix renaming parameter, which allows to customize the renaming of the prefixes while providing a unique way to normalize them. The process allows three renaming types according to the user's input parameter:

- ***Original Renaming***: allows the names of input prefixes to be preserved in the output RDF file. By default, in the case of two or more repeated namespaces with different prefixes, we preserve the shortest one. However, other preferences can be adopted as well (most significant one, most used, etc.).

  For instance, the original renaming of the namespaces in the use case 2 (Section 3.2.1) is:

  rdf = `http://www.w3.org/1999/02/22-rdf-syntax-ns#`

  dc = `http://purl.org/dc/elements/1.1/`

  ex = `http://example.org/stuff/1.0/`

- ***System Renaming***: generates prefixes using a default formal grammar[1] (Ω) (with terminal and non-terminal symbols, and a set of production rules defining the grammar's

[1]Moreover, we can customize the grammar w.r.t. user's requirements.

prefix language, L(Ω) [HMU01]) which is composed of: i) an alphabet of terminal symbols, ii) an alphabet of nonterminal symbols, iii) an initial symbol, iv) a set of production rules, and v) a number of repetitions, represented as:

$$\Omega = \{\Sigma_T, \Sigma_N, S, P, n\}$$

where:

$\Sigma_T = \{a, \ldots, z, A, \ldots, Z, 0, \ldots, 9\}$
$\Sigma_N = \{prefix, lowerletter, upperletter, digit, name, \_\}$
$S = prefix$
P is a set of production rules:

$< lowerletter > ::= a|b|\ldots|z$

$< upperletter > ::= A|B|\ldots|Z$

$< digit > ::= 0|1|\ldots|9$

$< prefix > ::= \_ < name >$

$< name > ::= [< lowerletter > \mid < upperletter > \mid < digit > [| < name >]]$

For instance, the system renaming of the namespaces in the use case 2 (Section 3.2.1) is:

_a = `http://example.org/stuff/1.0/`

_b = `http://purl.org/dc/elements/1.1/`

_c = `http://www.w3.org/1999/02/22-rdf-syntax-ns#`

- ***Collective Renaming*:** generates prefixes using an inverted index to store all the generated ones within a file collection and their corresponding namespaces. This also allows the generation of a collective index that could be shared among several users, which could later be beneficial in several scenarios (e.g., when the RDF descriptions have to be exchanged between multiple databases)[1].

  For instance, the collective renaming of the namespaces in the use case 2 (Section 3.2.1) may be:

  _a = `http://www.w3.org/1999/02/22-rdf-syntax-ns#`

  _b = `http://purl.org/dc/elements/1.1/`

  _c = `http://example.org/stuff/1.0/`

  Note that, in the collective renaming the order depends of the entry into the database, to generate the identifier.

[1]This will be investigated in a dedicated upcoming study.

4.5.2.2 Sorting Process (SP)

Normalization Rule 5 establishes node order variation, to have an appropriate and unique specification of the statements in the output serialization with respect to a sorting parameter $\tilde{p} :< iorder, sortc >$. The combination of the values of *iorde*r and *sortc* in parameter $\tilde{p}$ may vary according to the requirements of the user w.r.t. the targeted applications, using all possible triple orderings in *iorder* and the sorting criteria in *sortc*. For the *iorder*, we follow the six indexing schemes presented in [WKB08] (SPO, SOP, PSO, POS, OSP, OPS) describing the different combinations of the three elements composing an RDF statement (subject, predicate, object), and for *sortc*, we adopt *asc*, *des* and *null* to represent ascending, descending and no order respectively.

The sorting process is based in a ***Statement Sorting Expression*** (Ψ) which is composed of: i) an alphabet of terminal sorting symbols, ii) an alphabet of nonterminal sorting symbols, iii) an initial sorting symbol, iv) a set of production rules of the sorting, and v) a number of repetitions, represented as:

$$\Psi = \{\Sigma_{TS}, \Sigma_{NS}, IS, PS, n\}$$

where:

$\Sigma_{TS} = \{S, P, O, asc, desc, iri, bn, l\}$

$\Sigma_{NS} = \{order, index, element, type\_element, criteria\}$

$IS = order$

PS is a set of production rules:

$< element >::= s|p|o$

$< type\_element >::= iri|bn|l$

$< criteria >::= asc|desc|null$

$< order >::= [< index > \mid < index > \mid < index >]$

$< index >::= [< element > \mid < type\_element > \mid < criteria >]$

In this study, we assume the ascending (*asc*) order as a default value (used as reference to analyze data storage).

Our *Statement Sorting Expression* allows to reorder each element taking into account the type element and the criteria. Although our representation is generic, allowing to choose different order criteria for each (S, P, O) element, yet we simplify and consider that the same criterion will be chosen by the user for the three elements.

In Rule 5, we choose the SOP (subject-object-predicate) index as a default value since it: i) allows to group first the subject and object elements that describe the information of resources, and then the blank nodes. We adopted this approach since the number of different predicates is always much smaller than the number of different subjects or objects, which allows to perform sorting much faster. The improved efficiency of the SOP index was highlighted in [Fea13] and is reflected in our performance evaluation experiments in Section 6.4 (see Figure 6.8).

Taking into account our default parameter $\tilde{p}$, the sorting process is lexicographically ascending, based on the element type and on the values of subjects, objects, and predicates (SOP). The sorting is undertaken as follows:

- Reorder the statements according to the type of the subject (first IRI and after BN),
- Reorder the values of the subjects in lexicographic ascending order,
- For all subjects, reorder the statements according to the type of the object (literal, IRI and then BN),
- Reorder the values of the objects in lexicographic ascending order,
- Reorder the values of the predicates in lexicographic ascending order.

The pseudo-code of our statements sorting algorithm is provided in Algorithm 2. Note that sorting can be achieved in average linear time using efficient sorting algorithms such as *Quick Sort*, *Merge Sort*, *Bucket Sort* [Knu98]. We adopt a basic *Merge Sort* algorithm in our approach due to its constant complexity level (i.e., worst case O($N \times log(N)$) and average O(N) where N is the number of siblings being ordered). Details of our adapted *MergeSort* algorithm are provided in the Appendix (since it is widely known and used in practice), along with the algorithm describing our statement comparison operator ($\leq_\Psi$) defined following the statement sorting expression Ψ described above.

Algorithm 2 Statement Sorter

```
Input: $ST^{+}[]$ //List of Extended Statements of the RDF Description to be sorted
Output: $ST^{+'}[]$ //Sorted list of Extended Statements of the RDF Description
1: $ST^{+'} = MergeSort(ST^{+}[], \leq_\Psi)$ //where $\leq_\Psi$ is our statement comparison operator (cf. Algorithm 5)
2: return $ST^{+'}[]$
```

In addition, note that the statements' order has a direct impact in Web applications, e.g., in Jena Loading Time, the PSO index order has a better time performance in comparison with other indexes as we shown in Section 6.2.

4.5.2.3 Formatting Process (FP)

This component allows to: a) choose a specific form for the output RDF file, b) manage the variety of blank node serializations, and c) manage datatypes and languages[1].

Our current solution allows three different output forms (other forms could be devised based on user/application requirements):

- ***Flat*:** it develops each RDF statement one by one as a single declaration, i.e., each subject has one declaration in the file. In the case of blank node serialization, it uses nodeIds. For instance, Figure 3.3 shows a *flat* form output of the RDF graph in Figure 3.1.
- ***Compact*:** it nests the RDF statement, i.e., each statement may have another statement nested in its declaration. For the blank node serialization, this form uses the parse-Type="Resource". We show hereunder another serialization of the RDF graph in Figure 3.1 represented in *compact* form:

```
<?xml version="1.0" encoding="utf-8"?>
 <rdf:RDF xmlns:ex="http://example.org/stuff/1.0/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax
  -ns#">
 <rdf:Description rdf:about="http://www.univ-pau.fr">
  <ex:nameprof rdf:parseType="Resource">
   <ex:first_name rdf:datatype="http://www.w3.org/
   2001/XMLSchema#string">Sebastien</ex1:first_name>
   <ex:last_name xml:lang="fr">Durand</ex1:last_name>
  </ex:nameprof>
  <ex:lab rdf:resource="http://liuppa.univ-pau.fr/live/"/>
 </rdf:Description>
</rdf:RDF>
```

- ***Full compact*:** dedicated to RDF/XML format, it nests RDF statement, uses the ENTITY XML construct to reduce space by providing an abbreviation for IRIs[2], reuses the variables in the RDF file, and uses attributes instead of properties for the blank node serialization. We show hereunder yet another serialization of the RDF graph in Figure 3.1 using the *full compact* form:

[1]Not considering the cases when the datatypes and languages have different declarations in the statements.

[2]Refer to XML ENTITY construct in `http://www.w3.org/TR/xml-entity-names/`(IRI of XML W3C standard)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY _a 'http://liuppa.univ-pau.fr/'>
]>
<rdf:RDF
  xmlns:ex="http://example.org/stuff/1.0/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax
  -ns#">
<rdf:Description rdf:about="http://www.univ-pau.fr">
 ...
    <ex1:lab rdf:resource="&_a;live/"/>
  </rdf:Description>
</rdf:RDF>
```

Providing different output types is necessary to satisfy the requirements of different kinds of RDF-based applications. For instance, compact representations are usually of interest to human users when storing RDF data [AMMH07, CDD$^+$04, AÖD14, WW06, Wea03], and running and answering RDF queries [Gea11, HD05], yet less compact/more structured representations - which are easier to process by machines - could be useful in automated processing (e.g., automatic annotation of vector images into RDF files to be processed for image clustering/annotation recommendation [STC14]).

4.6 Summary

In this chapter, we proposed a syntactic RDF normalization process, as a means to transform RDF descriptions into a normalized representation in Section 4.5. To develop this approach, in Section 4.1, we presented definitions, functions and operators as the building blocks of our normalization process. Against this background, we reviewed relevant existing works in Section 4.2, highlighting the properties and limitations of solutions that authors proposed to solve the problems of RDF redundancies and disparities. Then, in Sections 4.3 and 4.4, we put forward a set of rules and properties to develop our approach in a formal and verifiable way.

Our approach allows to: i) preserve all the information in RDF descriptions, ii) eliminate all the logical redundancies and physical disparities in the output RDF description, iii) establish a unique specification of the statements in the RDF output description, iv) formalize the normalization process, and v) consider user parameters to handle the application requirements and adapt RDF output formats accordingly.

In Table 4.8, we show all the syntactic problems detailed in Sections 3.1 and 3.2, against the approaches detailed in Section 4.2 and our approach, to highlight and compare which problems were solved by each approach. Note that in the table 4.8: X means that the approach does not solve the problem, ✓* means that the approach solves partially the problem, and ✓

means that the approach solves the problem.

We extend this, in Chapter 5, by investigating other challenges pertaining to: semantic ambiguities and IRI discrepancies for the RDF normalization process described in Sections 3.3 and 3.4, developing an extension of the R2NR framework in order to perform semantic and IRI-aware RDF logical and physical normalization.

Table 4.8: Relation between problems and RDF syntactic Normalization approaches

Approaches	Problems										
	Syntactic Issues									Semantic Ambiguities	IRI Discrepancies
	P1	P2	P2.1	P2.2	P3	P4	P5	P6	P7		
Pathak et al. [Pea09]	X	X	X	X	X	X	X	X	X	X	✓*
Tao et al. [Tea09]	X	X	X	X	X	X	X	X	X	X	✓*
Jiang et al. [Jea13]	X	X	X	X	X	X	X	X	X	X	✓*
Belleau et al. [Bea08]	X	X	X	X	X	X	X	X	X	X	✓*
Hayes et al. [HG04]	✓	X	X	X	✓*	X	X	✓	X	X	X
Gutierrez et al. [Gea04, Gea11]	X	✓	X	X	X	X	X	X	X	X	X
Longley [Lon15]	X	✓	X	X	X	X	X	✓	X	X	X
Fernandez et al. [Fea13]	X	✓*	X	X	✓*	X	X	✓	X	X	X
Vrandecic et al. [Vea09]	X	X	X	X	X	X	X	X	✓	X	X
Our approach	✓	✓	✓	✓	✓	✓	✓	✓	✓	X	X

Chapter 5

Semantic and IRI RDF Normalization

"I think in general it's clear that most bad things come from misunderstanding, and communication is generally the way to resolve misunderstandings, and the Web's a form of communications, so it generally should be good."

— Tim Berners-Lee

As discussed in the previous chapter, RDF normalization has been treated for different approaches in knowledge representation, data integration, graph representation and syntax serialization (Section 4.2). However, all these approaches focus on syntactic problems disregarding other challenges related to semantic ambiguities and IRI coreference (Sections 3.3 and 3.4) that may also affect the RDF descriptions.

In this chapter, we present an extension of our RDF normalization approach by integrating solutions for logical redundancies and physical disparities that are caused by the presence of semantic ambiguities and IRI discrepancies in RDF descriptions. We first describe some functions (Section 5.1) developed for facilitating the understanding and creation of our normalization rules (Section 5.3). Next, we discuss related works regarding semantic ambiguity (Section 5.2.1), IRI identity (Section 5.2.2), and IRI coreference (Section 5.2.3) to understand the impact of these problems in the data duplication of RDF descriptions. Against this background, we discuss also the approaches related to RDF normalization w.r.t semantic and IRI problems (Section 5.2.4) which have influenced the understanding and design of our approach. We then develop our RDF Normalization extended approach with two additional levels: Semantic level and IRI level (Section 5.4). Finally, we conclude this chapter with a summary and a comparison between our approach with the approaches studied in this chapter (Section 5.5).

5.1 Normalization Functions

We start this chapter by providing functions related to our semantic and IRI normalization process, while reusing some of the functions and operators described in Sections 4.1.2 and 4.1.3. Table 5.1 summarizes the functions developed for the Semantic and IRI normalization.

Function **7 (Synonymy RDF selector *[Syn]*)** *The synonymy RDF selector function, noted* $Syn(st_i^+.o, st_j^+.o, KB)$*, takes as input two RDF objects of the respective extended statements and a knowledge base*[1] *(KB), and returns as output a boolean True or False value, designating whether they are synonyms or not*♦

For instance, in Figure 3.6.c given st_1^+ and st_2^+, where:

st_1^+: <`http://liuppa.univ-pau.fr/live`$_u$,ex:name$_\perp^+$, “LIUPPA”$_l$ >

st_2^+: <`http://liuppa.univ-pau.fr/live`$_u$,ex:name$_\perp^+$, “UPPA Computer Science Lab”$_l$ >

We can apply $Syn(st_1^+.o, st_2^+.o, KB)$ where:

$Syn(LIUPPA, UPPA\ Computer\ Science\ Lab, KB) = True$

Note that using Natural Language Processing (NLP) techniques such as acronym recognition and machine translation, we can recognize entities and their variants in different languages (e.g., “LIUPPA” is recognized as equivalent to “Laboratoire Informatique de l’UPPA” using acronym recognition, which in term is recognized as “UPPA Computer Science Lab” using machine translation).

Function **8 (Equivalent literals *[EquivLit]*)** *The equivalent literals function, noted* $EquivLit(st_i^+.o, st_j^+.o, KB, TP)$*, takes as input two RDF literals and two optional parameters: Knowledge base (KB) and Tolerance Parameter (TP)*[2]*, and returns as output a boolean True or False, designating whether the literals are equivalent or not. This function performs synonym detection (using function Syn) and data-type conversion using a dedicated function performing the necessary literal value type conversions (e.g., string to number, number to date, date to string, etc.) to evaluate whether literals are equivalent or not*♦

For instance, we show two examples:

a) In Figure 3.6.c given st_3^+ and st_4^+, where:

[1] A knowledge base is a structure resources as thesaurus, machine-readable dictionaries, or ontologies.

[2] The tolerance parameter allows the system to evaluated two numbers that are considered equivalent if their difference is less than the tolerance

st_1^+: <`http://liuppa.univ-pau.fr/live`$_u$,ex:name$_\perp^\perp$, “LIUPPA”$_l$ >

st_2^+: <`http://liuppa.univ-pau.fr/live`$_u$,ex:areaTotal$_\perp^\perp$, “UPPA Computer Science Lab”$_l$ >

We can apply $EquivLit(st_3^+.o, st_4^+.o, KB, null)$ where:

$EquivLit(25, 25.4, KB, null) = True$ given a KB.

b) In Figure 3.6.d given st_3^+ and st_4^+, where:

st_3^+: <`http://liuppa.univ-pau.fr/live`$_u$,ex:areaTotal$_{int}^\perp$, 25$_l$ >

st_4^+: <`http://liuppa.univ-pau.fr/live`$_u$,ex:areaTotal$_{decimal}^\perp$, 25.4$_l$ >

We can apply $EquivLit(st_3^+.o, st_4^+.o, null, TP)$ where:

$EquivLit(25, 25.4, KB, 0.5) = True$ given numerical tolerance 0.5.

Function 9 **(Semantic RDF selector *[SemSelect]*)** *The semantic RDF selector function, noted $SemSelect(st_i^+, st_j^+, TS, DH, LI)$, takes as input two RDF statements (for selecting only one) and three parameters (for giving the type of selection): Type Selector (TS) parameter (e.g., generic, short, specific, long), Datatype Hierarchy (DH) parameter (e.g., standard or user), Language Indicator (LI) parameter (e.g., user option: en, fr, etc., see Table 5.4 in Section 5.4), and returns as output one extended statement (st_{sem}^+) according to the parameters for semantic selection (blank nodes do not have datatypes nor languages, and thus will take null parameters when processed)*⧫

For instance, in Figure 3.6.c given st_1^+, st_2^+ and $TS =$ “*short*”, we can apply:

$SemSelect(st_1^+, st_2^+, TS, null, null)$ where:

$st_{sem}^+ = SemSelect(st_1^+, st_2^+, TS, null, null) = st_1^+$

Regarding the Type Selector (TS) parameter values: i) *generic* - *specific* designate whether more generic values or more specific values (namely data-types and/or literals) should appear in the output RDF description. For example, following the RDF/XML data-type hierarchy, “decimal” is considered as a more generic data-type compared with “int” (integer) which is more specific. Hence, a numeric value defined in one RDF/XML serialization as a decimal, or in another serialization as an integer, can be represented in the corresponding output normalized description following the user's choice of parameter TS[1]; ii) *short - long* highlight the preferred size of RDF elements in the output serialization. For instance, following the size of literals "LIUPPA" and "UPPA Computer Science Lab", "LIUPPA" is considered shorter as a literal value than "UPPA Computer Science Lab" which is longer. Note that we use default value "short" for parameter TS in our normalization process, which aims to reduce the output (normalized) RDF description's size. Yet, the selection of TS parameter values ultimately depends on user preference and the target application. The selection of TS parameter values ultimately depends on user preference and the target application.

Function **10 (Identify *[identify]*)** *The identify function, noted $identify(st_i^+.o, IL)$, takes as input an IRI object and the IRI Layer (IL) parameter (for giving the type of IRI analysis), and returns as output the identity of the IRI {identifier, document, document representation, ontology, concept}*♦

For instance, in Figure 3.11.b, given:

$st_1^+.o =$`http://it.dbpedia.org/resource/Lussemburgo` and $IL = network$

We can apply $identify(st_1^+.o, IL)$, where:

$identify($`http://it.dbpedia.org/resource/Lussemburgo`$, IL) = identifier$

Function **11 (Statement Selector *[StSelect]*)** *The statement selector function, noted $StSelect(st_i^+, st_j^+, TS)$, takes as input two RDF extended statements and a type selector (TS) parameter, and returns as output an equivalent RDF extended statement (st_{result}^+), to be later removed from the RDF description according to parameter TS.*

For instance, in Figure 3.11.b, given st_1^+, st_2^+ and TS = "*short*", where:

st_1^+: <`http://dbpedia.org/resource/Luxembourg`$_u$,owl:sameAs$_\perp^+$, `http://it.dbpedia.org/resource/Lussemburgo`$_u$ >

st_2^+: <`http://dbpedia.org/resource/Luxembourg`$_u$,owl:sameAs$_\perp^+$, `http://it.dbpedia.org/resource/Luxemburgo`$_u$ >

We can apply $StSelect(st_1^+, st_2^+, TS)$ and obtain:

[1]The parameter TS has the same behavior to evaluated the generality or specificity of literals values, blank nodes values or IRIs (id values, content, etc.).

$st^+_{result} = st^+_2$ Likewise, when we apply TS = "*long*", we obtain: st^+_1.

FUNCTION **12 (Namespace Selector *[NamespaceSelect]*)** *The namespace selector function, noted $NamespaceSelect(qn_i, qn_j, TS)$, takes as input two qnames and a type selector (TS) parameter, and returns as output an equivalent RDF qname (qn_{result}), to be later removed from the RDF description according to parameter TS.*

For instance, in Figure 3.8, given qn_1, qn_2 and TS = "*short*", where:

$qn_1 < px_1, ns_1 > \Longrightarrow px_1$ = "*dcterm*", ns_1 = "*http://schema.org/*"

$qn_2 < px_2, ns_2 > \Longrightarrow px_2$ = "*dct*", ns_2 = "*http://purl.org/dc/terms*"

We can apply $NamespaceSelect(qn_1, qn_2, TS)$ and obtain:

$qn_{result} = qn_1$ Likewise, when we apply TS = "*long*", we obtain: qn_2.

Table 5.1: Summarized descriptions of functions based on definition of the extended normalization process

Function	Input	Output	Description
Syn	$st^+_i.o$, $st^+_j.o$, KB	T or F	Returns a boolean value to designate whether the input values are synonyms or not
$EquivLit$	$st^+_i.o$, $st^+_j.o$, KB, TP	T or F	Returns a boolean value to designate whether the input values are equivalents or not
$SemSelect$	st^+_i, st^+_j, TS, DH, LI	st^+_{sem}	Returns an extended statement according to the parameters (*TS,DH,LI*)
$identify$	$st^+_i.o$, IL	$identity$	Returns the identity of the IRI (e.g., identifier, document, etc.)
$StSelect$	st^+_i, st^+_j, TS	st^+_{result}	Returns an extended statement according to the parameter *TS*
$NamespaceSelect$	qn_i, qn_j, TS	qn_{result}	Returns an qname according to the parameter *TS*

5.2 Related Work

The need for RDF normalization has been identified and discussed in various domains, ranging over domain-specific knowledge representation, data integration, as well as service and semantic data mediation. Yet, few existing studies have specifically addressed the issues of logical (graph) and physical (syntax) RDF normalization, also the need for solving the ambiguity and identity of a resource has been identified and discussed in various studies, e.g., text summarization [MA12], sub-structures for document summarization [LGMF04], semantic matching [ACM10], etc. Yet, few of them have specifically addressed the issues of the semantic ambiguity, IRI identity, and IRI coreference to perform RDF normalization.

For clarity of presentation, we classify state of the art methods in three categories: i) Semantic Disambiguation (briefly describing traditional semantic disambiguation for flat text and some approaches based on RDF descriptions), ii) IRI identity identification (methods to unambiguously identify resources in RDF descriptions), and iii) IRI coreferencing (methods that handle co-referencing through IRI disambiguation).

5.2.1 Resolving Semantic Ambiguity

Several approaches have been developed to manage the lexical ambiguity problem, the main problem is to identify: a term may have a multiple meanings (polysemy), a word can be implied by other related terms (metonymy), and/or several terms can have the same meaning (synonymy) [KC92, Tek16]. Also, in RDF context, literals and RDF resource names can be ambiguous and have multiple senses, this is a challenge for automated methods presented in [VKM07, MJB12, AAD$^+$09, VAGS06]. In [AAD$^+$09, VAGS06], the authors proposed disambiguating metadata of some ontologies (Gene Ontology, MeSH and RDF/OWL of WordNet) using sense disambiguation techniques. Authors in [VKM07] used techniques of name disambiguation to identify geographic features through the names. As Dbpedia is a big and important knowledge base of Linked Data in [MJB12], the authors used named entity recognition to extend the dataset.

All these aspects are treated in the Word Sense Disambiguation (WSD) approaches [Nav09] following four main elements: i) selection of word senses, ii) using external knowledge sources, iii) identifying the context, and iv) selection of an automatic classification method.

5.2.1.1 Selection of word senses

Word sense can not be easily discretized because the language is inherently subject to change and interpretation [Nav09]. In the literature, we find two possible solutions to select words for disambiguation [Tek16]: i) *all-words*, or ii) *lexical-example*. In [NWC03, PLDP07] the system is expected to disambiguate all the words in a flat document. For the *lexical-example* in [PLDP07], we found specific target words that are selected for disambiguation. Experimental results reported in [Nav09] show high disambiguation accuracy using the *lexical-example* approach against with the *all-words* approach. But, the major difficulty for adopting the *lexical-example* is the supervised learning approaches of selecting the target words (these approaches are time-consuming and requires training data which is not always available).

5.2.1.2 Using External Knowledge Sources

Knowledge is a main component of WDS, providing data which are needed to associate sense with words. The WSD methods can be distinguish as: i) corpus-based (unstructured resources) or ii) knowledge-base (structured resources) depending on the knowledge source they rely on [Tek16]. In [$M^{+}07$, Ped06], authors develop corpus-based (data-driven) approaches, involving information about words previously disambiguated, and require supervised learning from sense-tagged corpora (e.g., OntoNotes [$PRM^{+}11$] and SemCor [MLTB93]) where words have been associating with explicit semantic meaning, in order to enable predictions for few words. On the other hand, in [Mih06, NV05] knowledge-based (knowledge-driven) methods handle a structured sense inventory and/or repository of information about words and in this way they can automatically distinguish their meanings in the text. The structures resources (knowledge base) as thesaurus (e.g., Roget's thesaurus [Yar92]), machine-readable dictionaries (e.g., WordNet [Mil95] and Yago [HSBW13]), and ontologies (e.g., FOAF [BM12]) provide ready-made sources of information about word senses.

5.2.1.3 Identifying the context

Another important issue in WSD is to identify the context of the words selected to be disambiguated. In fact, the context give us more meaning and information about the words that we need to disambiguate. In traditional textual data, the context of a word usually consist of the set of terms in the word's vicinity, i.e., terms occurrence close to the word, within a certain predefined window size [Les86]. After the context is identified, it has to be effectively represented to perform disambiguation computation [Tek16]. The literature shows different methods to determine the co-occurrence of the words in flat text ([IV98]) and structured documents ([AMdLS06]).

5.2.1.4 Selection of an automatic classification method

The final step is how we can associate the sense with words, taking into account the three first points mentioned before. We can broadly distinguish two main approaches to WSD: i) supervised WSD and ii) unsupervised WSD. One hand, supervised methods use machine-learning techniques to learn a classifier from labeled training sets. On the other hand, unsupervised methods are based on unlabeled corpora, not requiring any human interaction (completely automated). Most recent (and RDF-related) approaches, e.g., [VKM07, MJB12, AAD+09, VAGS06, LGMF04, MA12], make use of a machine-readable knowledge based (e.g., WordNet [Mil95]).

Many approaches have been proposed for flat text [Nav09] and semi-structured XML-based data [Tek16], but very few approaches have been dedicated for RDF and linked data disambiguation.

In the remainder of this study, we constrain our proposal to the general definition of unsupervised WSD using a reference KB (i.e., unsupervised and knowledge-based WSD) [Nav09] because corpus-based and supervised methods are highly time consuming and require extensive and reliable training sets (to produce a relevant sense-annotated corpus) which are not always available, while knowledge-base and unsupervised methods do not require any human interaction or training phase, reducing the time consumption and automating the obtention of meanings of the words in the text.

5.2.2 Resolving IRI Identity

The Semantic Web introduces a problem with IRI identity, having into account: the meaning of a resource, and how it is represented. The Semantic Web introduces a problem with the IRI identity, having to do with the meaning of a resource and how it is referred to. Here, the diversity in different types of IRIs (cf. Figure 2.4), referring to both information and non-information resources, highlights a so-called "identity crisis" [BSMG06]: how to distinguish the identities of IRIs referring to information resources (on the Web), and especially IRIs referring to non-information resources (on the Semantic Web). For instance, how can we identify which of the following non-information IRIs is actually referring to the required information pertaining to Sebastien: `http://www.example.com/id/sebastien` and `http://www.example.com/doc/sebastien`. To address the problem, various methods have been suggested [AV08, Hal11, Hal13b, JGM07, BSNM08, BSB08, Boo08, HH08, HHM+10, HH10, HPUZ10], which we organize in two main categories, i.e., methods performing IRI identification: i) at the network layer, and ii) at the data layer, based on user required services (cf. Figure 5.1).

5.2.2.1 Network layer

Methods in this category, e.g., [AV08, Hal11, Hal13b, JGM07, BSNM08, BSB08], utilize the HTTP protocol to identify the Semantic Web IRIs. Here, two solutions have been suggested: using Hash IRIs and using 303 IRIs[1] [Fie03]. Methods of the former group use the hash symbol (#) to fragment the IRI, separating its so-called root from its definition, e.g., IRI

[1]urlhttp://lists.w3.org/Archives/Public/www-tag/2005Jun/0039.html

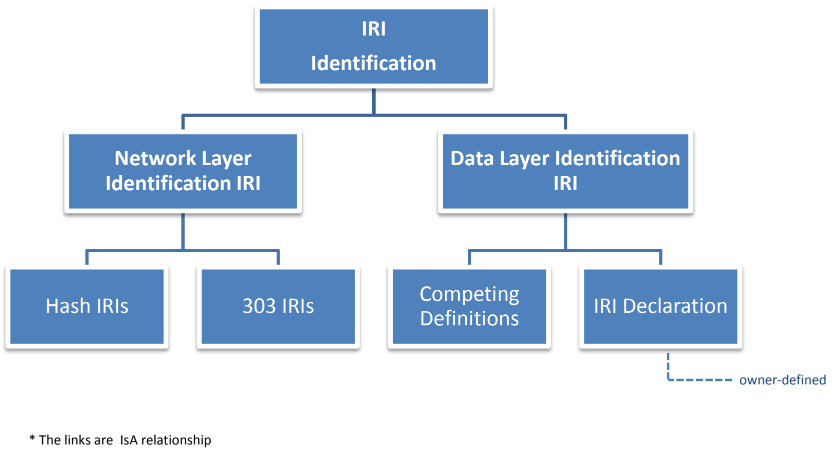

IRI Identification
Network Layer Identification IRI
Data Layer Identification IRI
Hash IRIs
303 IRIs
Competing Definitions
IRI Declaration
owner-defined
* The links are IsA relationship

Figure 5.1: Taxonomy of IRI identification methods

`http://www.example.com/about#sebastien` would be fragmented into the root `http://www.example.com/about` and the definition *sebastien* (cf. Figure 5.2). The Hash IRI approach is generally preferred when handling small and stable sets of resources that evolve together (e.g., RDF schema vocabularies and OWL ontologies) [BG14, PSHH$^+$04], as it reduces the number of unnecessary HTTP round-trips and consequently access latency, while allowing IRIs to share the same non-hash part. Yet, the major drawback of this technique is the need for loading the data for all resources sharing the same root, because they are all in the same physical file (location).

With 303 IRI solution, IRI identification is handled using a special HTTP status code: 303 See Other HTTP header [Fie03], which allows to indicate that the requested resource is not an information IRI (i.e., it is not a Web document) by dereferencing the IRI itself to obtain a new IRI, which can in turn be dereferenced, until reaching an IRI definition. Dereferencing can happen by going through i) one so-called generic document which then links to others, or ii) by linking directly to different documents [Hal13b]. For example, in Figure 5.3, IRI `http://www.example.com/id/sebastien` is dereferencing to on generic document referred to by `http://www.example.com/doc/sebastien`, which is in turn dereferenced to different documents: an RDF document (`http://www.example.com/doc/sebastien.rdf`) and an HTML page (`http://www.example.com/doc/sebastien.html`). However, Figure 5.4 shows the same IRI (`http://www.example.com/id/sebastien`) which directly derives to the RDF document or HTML page without using a generic document. The 303 IRI method is usually more suitable when dealing with large sets of data (e.g., RDF owner descriptions). In addition, with the 303 IRI technique, the redirection target can be configured separately for each resource, hence reducing network delay. This technique is also flexible with respect to the Hash IRI method because it considers two dereferencing approaches: using generic documents or different documents, which is not allowed (and can be coupled) with Hash IRI. While effective, yet the 303 IRI technique can produce a large number of redirects, thus ultimately causing high network

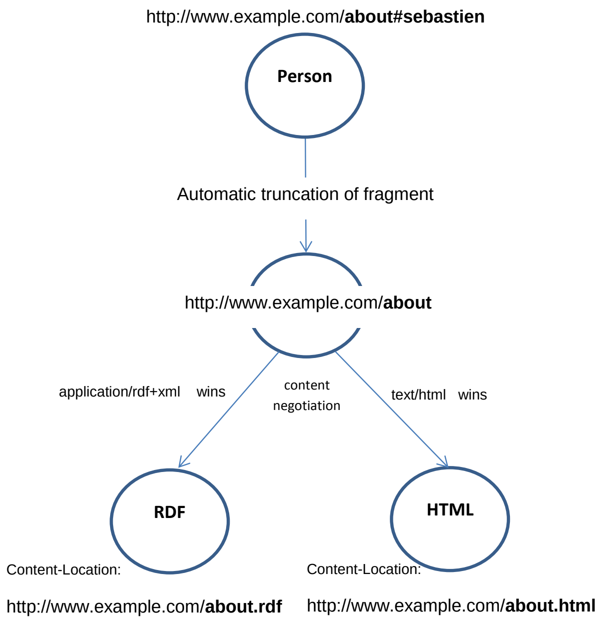

http://www.example.com/about#sebastien
Person
Automatic truncation of fragment
http://www.example.com/about
application/rdf+xml wins
content negotiation
text/html wins
RDF
HTML
Content-Location:
http://www.example.com/about.rdf
Content-Location:
http://www.example.com/about.html

Figure 5.2: Example using Hash IRIs

latency, and may even require downloading all data to process a large number of requests in a timely manner.

5.2.2.2 Data layer

Methods based on the data layer, e.g., [Boo08, HH08, HHM$^+$10, HH10, HPUZ10], can also be organized in two categories: i) using competing definitions, and ii) using IRI declaration. Methods of the former category, e.g., [MJB12, SHJJ09, HHM$^+$10, HH10, HPUZ10, Boo08], assume that all RDF statements are created equal, so the community or marketplace decides which statements become the prevailing definition of a particular IRI, e.g., st_1: <`http://dbpedia.org/resource/Luxembourg`, *owl:sameAs*, `http://es.dbpedia.org/resource/Luxemburgo`> and st_2: <`http://dbpedia.org/resource/Luxembourg`, *wdrs:describedBy*, `http://dbpedia.org/data/Luxembourg.nt`> where *sameAs* and *describedBy* are the competing definitions. Yet, methods of the latter category (using IRI declarations), e.g., [Boo08, HH08, HPUZ10], assume that RDF statements are not created equal: some are special from the outset (core assertions). Here, the use of IRIs becomes less straightforward identifying the prevailing definition using a follow-your-nose strategy [Boo08] (considering all statements containing the IRI), such that the IRI should be consistently used as the definition in all statements. For example, considering st_1: <`http://www.univ-pau.fr`, *ex1:lab*, `http://liuppa.univ-pau.fr/live/`>, predicate *lab* can have a definition provided by the owner (i.e., creator of the statement), e.g., a laboratory. As a result, competing definitions become more ambiguous, since an IRI can mean

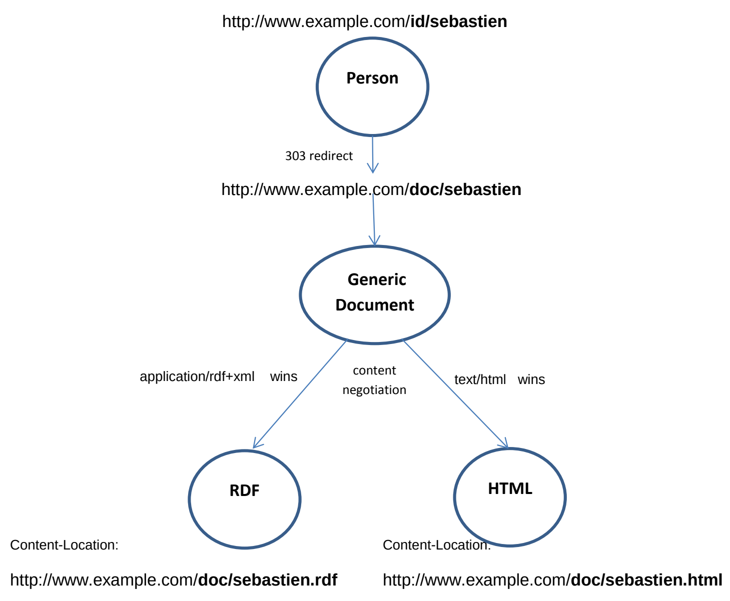

http://www.example.com/id/sebastien
Person
303 redirect
http://www.example.com/doc/sebastien
Generic Document
application/rdf+xml wins
content negotiation
text/html wins
RDF
HTML
Content-Location:
http://www.example.com/doc/sebastien.rdf
Content-Location:
http://www.example.com/doc/sebastien.html

Figure 5.3: Example using Hash IRIs forwarding to one Generic Document

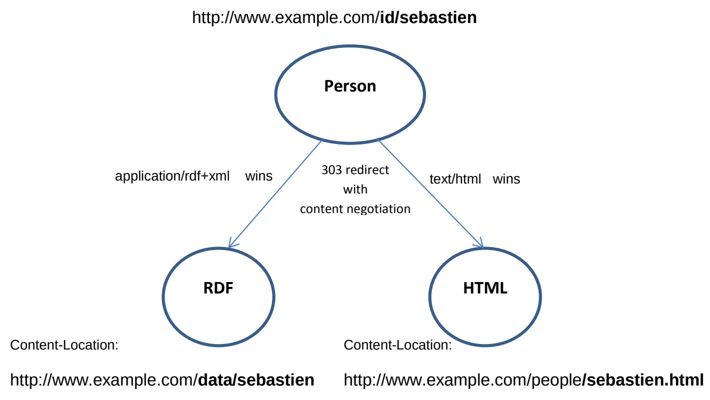

http://www.example.com/id/sebastien
Person
application/rdf+xml wins
303 redirect with content negotiation
text/html wins
RDF
HTML
Content-Location:
http://www.example.com/data/sebastien
Content-Location:
http://www.example.com/people/sebastien.html

Figure 5.4: Example using Hash IRIs forwarding to different documents

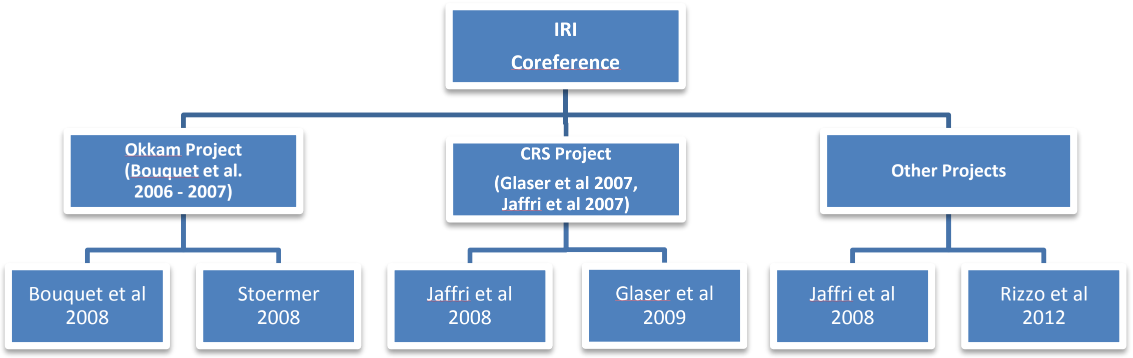


Figure 5.5: Taxonomy of IRI coreference methods

what the owner chooses it to mean, or what the user would like it (perceives it) to mean[1].

Hence, in most approaches for entity name systems [JGM07, BSB08] and summarization [MA12, LGMF04] in the literature, the competing definitions approach is adopted, allowing the user community to agreed upon the prevailing IRI definition, thus facilitating the sharing of unambiguous information without complications.

5.2.3 Handling IRI Coreference

As the Semantic Web is an open place to publish the information, it is inevitable to have multiple IRIs that reference the same resource. Hence, several methods were developed to address the IRI coreference problem, e.g., [BSMG06, BSG07, BSB08, BSNM08, GLMD07, JGM07, JGM08a, GJM09, JGM08b, RT12, LAH$^+$09], in order to help Semantic Web applications refactor and/or republish the data. In this context, most existing approaches are based on one of two main initiatives (cf. Figure 5.5):

- The first initiative is OKKAM's approach [BSMG06, BSG07] that advocates universally agreed IRIs for each entity with a centralized system. The goal of the OKKAM project, is not only to create a naming service for (non-information) resources, but also to create a directory containing resource profiles under the single control of one authority. Additionally, this approach has a service called OKKamCore that allows to modify, remove and publish resources and RDF statements based on a set of criteria [BSMG06]. Based on this project, the authors in [BSB08, BSNM08, Sto08] suggested the development of an Entity Name System (ENS) as a Web service to provide unique and uniform names for resources.
- The second initiative is the CRS approach [GLMD07, JGM07] that provides a service

[1]The IRI owner naming can have a lot of restrictions and problems in the implementation, because each IRI will vary based on each owner

which allows the publication and maintenance of coreference information in a single store with a distributed system. In this way, clients can discover alternative IRIs to the one they are using and use this information to help them for creating relations, RDF triples, graphs, etc. Additionally, this approach has a general service called SameAs.org[1] which provides coreference data in RDF formats. Based on the CRS project, the authors in [JGM08a, GJM09] propose to manage equivalent IRIs referring to the same concept or entity (IRI synonymity), analyzing the advantages and disadvantages of using the CRS proposal over other coreference methods.

Some authors propose to use OKKAM or CRS interchangeably [JGM08b], or suggest to utilize alternative services for extracting (non-information) resources[2] as Named Entity extraction techniques for extracting, classifying and disambiguating named resources [LAH+09] and provide IRI disambiguation for each entity that was extracted from a specific text [RT12] using keywords provided by users.

In light of the above presentation and discussion, we will focus on unsupervised WSD using a reference KB for solving semantic ambiguities in RDF statements (due to their reduced execution time and usually improved quality in comparison with unsupervised approaches [Tek16]), while combining network layer and data layer IRI identification (by taking into account the pros of both strategies), as well as using the CRS based SameAs service for solving IRI coreferences (which seems generally more robust and efficient that OKKAM's approach [JGM08b]). Recall that our ultimate goal remains to perform RDF normalization: i.e., to remove all the logical redundancies and physical disparities in RDF descriptions which can exacerbated with the presence of semantic ambiguities and IRI identify and coreference discrepancies as we develop in the sections 3.3 and 3.4.

5.2.4 Semantic and IRI RDF Normalization

5.2.4.1 Semantic Mediation

Semantic interoperability between RDF stores is also becoming an essential requirement: allowing different systems to communicate "meaningfully" with each other, exchanging RDF data and services despite the heterogeneous nature of the underlying information structures. Semantic interoperability can be achieved by the development of comprehensive shared information models using SW technologies (e.g., shared RDF or OWL reference ontologies defining common semantics following the SW vision)[KVS07, GCGP10], or by providing appropriate semantic mediators (translators), in order to convert information following the data format which each

[1]http://sameas.org/

[2]http://www.alchemyapi.com, http://dbpedia.org/spotlight, http://extractiv.com, http://opencalais.com, http://zemanta.com, http://www.sindice.com/

system understands [Kea08, Kea07, ZV11]. Table 5.2 shows the summary of all the approaches presented in this section.

Table 5.2: Summarized semantic mediation approaches

App.	Data Targeted	Features	Limitations	Aplication		Output
				Domain	Area	
Krogstie et al. [KVS07]	OWL	• Defining common semantics following the Semantic Web vision • Trying to produce a universal ontology	Blank_nodes Literals Statements Namespaces	Not specified	Semantic Interoperability	OWL
Garcia -Castro et al. [GCGP10]	OWL, RDF/XML	• Identifying ontology heterogeneity levels: lexical, syntactic, paradigm, terminological, conceptual, and pragmatical • Storing ontologies and sharing resources (URI)	Blank_nodes Literals Statements Namespaces	Not specified	Semantic Interoperability and Benchmarking	OWL
Kerzazi et al. [Kea07]	RDF file	Proposal name: Ontology - Based mediator • The result of the RDF data and query are normalized based on integrated ontologies • Using matching techniques • Registration of resources' semantics by relating them with ontologies	Blank_nodes Literals Statements Namespaces	Not specified	Data Integration	RDF file
Kerzazi et al. [Kea08]	Query RDF	Using the Ontology - based mediator in [Kea07] • Translating and optimizing the query • Using several phases into an executable query plan	Blank_nodes Literals Statements Namespaces	Not specified	Data Integration	Plain text

In most of the above mentioned projects, RDF normalization is viewed as applying predefined mapping dictionaries to define the correspondences between the original data constructs and the RDF constructs. In other words, most studies consider the original data to be well organized (normalized), thus the resulting RDF data would allegedly follow. Note that in most of these projects, issues of redundancies in RDF logical and syntax representations, which can occur in the produced RDF descriptions, are mostly left unaddressed.

5.2.4.2 IRI Disambiguation in RDF descriptions

Resource disambiguation in RDF descriptions is also becoming a challenge to normalized RDF descriptions, allowing to reduce number of IRIs and reuse them in the LOD context. Several approaches have been developed to IRI disambiguation, using different datasets as DBpedia, DBLP, WordNet, OpenCyc[1], etc. in [JGM08b, RŠD+10, SNA12, UNR+14]. In [JGM08b], the authors address the problem of coreference and provides an analysis about two main solutions for IRI disambiguation. The first one, ReSIST [Srl] project that has gathered metadata from publications and institutions and exposed them as linked data, using 15 repositories with their own CRS [GLMD07, JGM07], each CRS can use different algorithms to identify equivalent resources (see Section 5.2.3). The second one, OKKAM project, detailed also in Section 5.2.3, it is considered as a generator of "Web Entities" where the main aims are to create a naming service for resources and a directory with resource profiles.

For service-oriented natural language processing, there are some approaches that use SW resources (RDF/OWL) to disambiguate text [ŠRD+09, RŠD+10], thus providing annotations of words and improving the semantic graph quality, by merging nodes that refer to the same disambiguated concept. In a related study [UNR+14] to resource disambiguation, the authors introduce AGDISTIS as a knowledge base approach for named entity disambiguation. AGDISTIS combines the *Hypertext-Induced Topic Search* (HITS) algorithm with string similarity measures (breadth-first) and label expansion strategies to detect the correct IRI for a given set of named resources.

In [SNA12], in order to disambiguate a resource and generate an automatic query segmentation, the authors leverage the semantic relationships between data items using Markov models and HITS algorithm to disambiguate resources. Results of this approach show that it is robust with regard to query expression variance of resources.

Note that, in most of the above mentioned projects, RDF normalization is viewed as disambiguating IRIs. In other words, the focus of these studies are in handeling coreference between IRIs, to obtain the correct IRI (normalized) for a resource. Finally, issues of redundancies and disparities generated for IRI coreference are left unaddressed. Table 5.3 shows the summarization of all the approaches presented in this section.

5.3 RDF Normalization Rules

In this section, we provide a set of rules to resolve the normalization problems generated by semantic ambiguities and IRI discrepancies listed in Section 3.

5.3.1 Solving Logical Redundancies generated by Semantic Ambiguities

Given an input RDF graph G, logical redundancies related to node duplication based on semantic ambiguities (presented in Section 3.3.3) can be eliminated from G by applying the following

[1]`http://sw.opencyc.org`

Table 5.3: Summarized Resource disambiguation approaches

App.	**Data Targeted**	**Features**	**Limitations**	**Aplication**		**Output**
				Datasets	**Area**	
Jaffri et al. [JGM08b]	RDF description	• Analysing two solutions to the coreference problem • ReSIST [Srl] handles coreference from publications and institutions with 15 repositories each with their own CRS [GLMD07, JGM07] • OKKAM project [BSMG06, BSG07] is a centralised system to create a naming service for entities	Blank_nodes Literals Statements Namespaces	DBLP DBpedia	Identity management	Theoretical approach
Rusu et al. [RŠD+10]	Text fragment	• Extension of Enrycher [ŠRD+09] with RDF/OWL word sense annotation • Every word or collocation in a text fragment is annotated with the correspond resources to WordNet and OpenCyc	Blank_nodes Literals Statements Namespaces	WordNet OpenCyc	Service-oriented NLP	Not specified
Shekarpour et al. [SNA12]	n-tuple	• Automatic query segmentation and resource disambiguation method leveraging background knowledge • Leveraging the semantic relationship between data items using Markov models • Distributing a normalized connectivity degree across the state space (with HITS algorithm)	Blank_nodes Literals Statements Namespaces	DBpedia	Data Integration	RDF file
Usbeck et al. [UNR+14]	N3 Text	Proposal name: AGDISTIS • Named entity disambiguation approach and framework • Combining HITS algorithm with label expansion and string similarity measures	Blank_nodes Literals Statements Namespaces	Reuters-21578 RSS-500 news.de AIDA-YAGO2 MSNBC AQUAINT IITB	Information Extration	Plain text (HITS)

transformation rules:

Rule 1 - Semantic Statement Elimination containing blank node elements: It is designed to eliminate node duplications with blank node elements using the semantic select function ($SemSelect$) between extended statements. More precisely, $\forall st_i^+, st_j^+ \in ST^+(G)$ / $i \neq j$, if $st_j^+ \preceq st_i^+ \Longrightarrow remove((st_{sem}^+ \cup O(st_{sem}^+)), ST^+(G))$, where:

$st_{sem}^+ = SemSelect(st_i^+, st_j^+, TS, null, null)$.

Rule 2 - Semantic-based Statement Elimination containing literal elements: It is designed to eliminate node duplications with literal nodes using the semantic select function ($SemSelect$) between extended statements. More precisely, $\forall st_i^+, st_j^+ \in ST^+(G)$ / $i \neq j$, if $st_j^+.s = st_i^+.s \wedge st_j^+.p = st_i^+.p \wedge st_i^+.to = st_j^+.to = \text{“}literal''\wedge Syn(st_i^+.o, st_j^+.o, KB) \Longrightarrow remove(st_{sem}^+, ST^+(G))$, where:

$st_{sem}^+ = SemSelect(st_i^+, st_j^+, TS, null, null)$.

Note that literals can be affected by properties as datatypes and languages. Considering these properties, we establish variants of the Rule 2 as follows:

Rule 2.1 - Semantic-based Statement Elimination with same datatype: It is designed to eliminate node duplications when literals are affected by the same datatype using the semantic select function ($SemSelect$) between extended statements. More precisely, $\forall st_i^+, st_j^+ \in ST^+(G)$ / $i \neq j$, if $st_j^+.s = st_i^+.s \wedge st_j^+.p = st_i^+.p \wedge st_j^+.to = st_i^+.to = \text{“}literal'' \wedge st_j^+.dt = st_i^+.dt \wedge EquivLit(st_i^+.o, st_j^+.o, KB, TP) \Longrightarrow remove(st_{sem}^+, ST^+(G))$, where:

$st_{sem}^+ = SemSelect(st_i^+, st_j^+, TS, null, null)$

Rule 2.2 - Semantic-based Statement Elimination with different datatypes and same object values: It is designed to eliminate node duplications when literals are affected by different datatypes while having the same object (literal) using the semantic select function ($SemSelect$) between extended statements. More precisely, $\forall st_i^+, st_j^+ \in ST^+(G)$ / $i \neq j$, if $st_j^+.s = st_i^+.s \wedge st_j^+.p = st_i^+.p \wedge st_j^+.o = st_i^+.o \wedge st_j^+.to = st_i^+.to = \text{“}literal'' \wedge st_j^+.dt \neq st_i^+.dt \Longrightarrow remove(st_{sem}^+, ST^+(G))$, where:

$st_{sem}^+ = SemSelect(st_i^+, st_j^+, TS, DH, null)$

Rule 2.3 - Semantic-based Statement Elimination with different datatypes and different object values: It is designed to eliminate node duplications when different literals, having the same meaning, are affected by different datatypes using the semantic select function ($SemSelect$) between extended statements. More precisely, $\forall st_i^+, st_j^+ \in ST^+(G)$ /

$i \neq j$, if $st_j^+.s = st_i^+.s \wedge st_j^+.p = st_i^+.p \wedge st_j^+.to = st_i^+.to =$ "*literal*" $\wedge\ st_j^+.dt <> st_i^+.dt \wedge EquivLit(st_i^+.o, st_j^+.o, KB, TP) \Longrightarrow remove(st_{sem}^+, ST^+(G))$, where:

$$st_{sem}^+ = SemSelect(st_i^+, st_j^+, TS, DH, null)$$

Rule 2.4 - Semantic-based Statement Elimination with same language tag: It is designed to eliminate node duplications when literals are assigned the same language tag, using the semantic select function (*SemSelect*) between extended statements. More precisely, $\forall st_i^+, st_j^+ \in ST^+(G)\ /\ i \neq j$, if $st_j^+.s = st_i^+.s \wedge st_j^+.p = st_i^+.p \wedge st_j^+.to = st_i^+.to =$ "*literal*″ $\wedge\ st_j^+.lang = st_i^+.lang \wedge Syn(st_i^+.o, st_j^+.o, KB) \Longrightarrow remove(st_{sem}^+, ST^+(G)$, where:

$$st_{sem}^+ = SemSelect(st_i^+, st_j^+, TS, null, null)$$

Rule 2.5 - Semantic-based Statement Elimination with different language tags and same object value: It is designed to eliminate node duplications when literals are affected by different language tags using the semantic select function (*SemSelect*) between extended statements. More precisely, $\forall st_i^+, st_j^+ \in ST^+(G)\ /\ i \neq j$, if $st_j^+.s = st_i^+.s \wedge st_j^+.p = st_i^+.p \wedge st_j^+.to = st_i^+.to =$ "*literal*″ $\wedge\ st_j^+.o = st_i^+.o \wedge st_j^+.lang \neq st_i^+.lang \Longrightarrow remove(st_{sem}^+, ST^+(G)$ where:

$$st_{sem}^+ = SemSelect(st_i^+, st_j^+, TS, null, LI)$$

Rule 2.6 - Semantic-based Statement Elimination with different language tags and different object values: It is designed to eliminate node duplications when different literals, having the same meaning, are affected by different language tags using the semantic select function (*SemSelect*) between extended statements. More precisely, $\forall st_i^+, st_j^+ \in ST^+(G)\ /\ i \neq j$, if $st_j^+.s = st_i^+.s \wedge st_j^+.p = st_i^+.p \wedge Syn(st_i^+.o, st_j^+.o, KB) \wedge st_i^+.to = st_j^+.to =$ "*literal*″ $\Longrightarrow remove(st_{sem}^+, ST^+(G))$ where:

$$st_{sem}^+ = SemSelect(st_i^+, st_j^+, TS, null, LI)$$

Recall that we use default value "short" for parameter TS in our normalization rules, which aims to reduce the output (normalized) RDF description's size. Yet, any other TS parameter value can be chosen by the user, following her/his preferences and the target application.

5.3.2 Solving Logical Redundancies generated by IRI Discrepancies

Given an input RDF graph G, logical redundancies related to node duplication, and edge duplication based on IRI discrepancies (presented in Sec. 3.4.4) can be eliminated from G by applying the following transformation rules:

Rule 3 - IRI-based Statement Elimination: It is designed to eliminate edge duplications and/or node duplications using the entity select function ($StSelect$) between extended statements, which we detail in the following in two variants:

Rule 3.1 - IRI-based Statement Elimination based on objects: It is designed to eliminate node duplications using the entity select function ($StSelect$) between extended statements. More precisely, $\forall st_i^+, st_j^+ \in ST^+(G)$ / $i \neq j$ if $identify(st_i^+.o, IL) = identify(st_j^+.o, IL)) \wedge st_i^+.s = st_j^+.s \wedge st_i^+.p = st_j^+.p \Longrightarrow remove(st_{result}^+, ST^+(G)) \wedge replace(st_{result}^+.o, st_{select}^+.o, ST^+(G))$, where:

$st_{result}^+ = StSelect(st_i^+, st_j^+, TS)$ and $st_{result}^+.o$ is the IRI selected as equivalent from the extended statement st_{result}^+ and $st_{select}^+.o$ is the IRI selected (i.e., equivalent) of the other extended statement.

Rule 3.2 - IRI-based Statement Elimination based on predicates: It is designed to eliminate edge duplications using the entity select function ($StSelect$) between extended statements. More precisely, $\forall st_i^+, st_j^+ \in ST^+(G)$ / $i \neq j$ if $identify(st_i^+.p, IL) = identify(st_j^+.p, IL)) \wedge st_i^+.s = st_j^+.s \wedge st_i^+.o = st_j^+.o \Longrightarrow remove(st_{result}^+, ST^+(G))$, where:

$st_{result}^+ = StSelect(st_i^+, st_j^+, TS)$

5.3.3 Solving Physical Disparities generated by IRI Discrepancies

Given an input RDF file F, physical disparities related to namespace duplication base on IRI discrepancies (presented in Section 3.4.5) can be eliminated from F by applying the following transformation rule:

Rule 4 - IRI-based Namespace Duplication Elimination: It is designed to eliminate namespace duplications using the namespace select function ($NamespaceSelect$) in an RDF file F. More precisely: $\forall qn_i, qn_j \in QN(F)$ / $i \neq j$ if $identify(qn_i.ns, IL) = identify(qn_j.ns, IL)) \Longrightarrow remove(qn_{result}, QN(F)) \wedge replace(qn_{result}.ns, qn_{select}.ns, ST^+(F))$, where:

$qn_{result} = NamespaceSelect(qn_i, qn_j, TS)$ and qn_{select} select (i.e., equivalent) namespace.

5.4 RDF Normalization Process

Here, we build on the R2NR architecture provided in Chapter 4 to consider all the new semantic and IRI-based problems presented in Section 3 (i.e., semantic ambiguities between literals and blank nodes, as well as IRI identification and coreference discrepancies). We develop the overall architecture of our new R2NRE (RDF to Normalized RDF Extended) framework, depicted in Figure 5.6. It consists of three levels: i) syntactic level, ii) semantic level and, iii) Entity level with one extra module called IRI handler. In short, the three levels have different algorithms to remove and manage the redundancies and disparities discussed in Section 3, based on the normalization rules developed in Section 5.3. R2NRE accepts as input an RDF graph or an RDF file as well as a set of parameters (summarized in Table 5.4) to handle the semantic and IRI-based normalization according to the requirements of the user (cf. detailed descriptions of functions in Section 5.1 and operator parameter in Section 4.1.3). In the following, we describe the semantic and IRI level in more details.

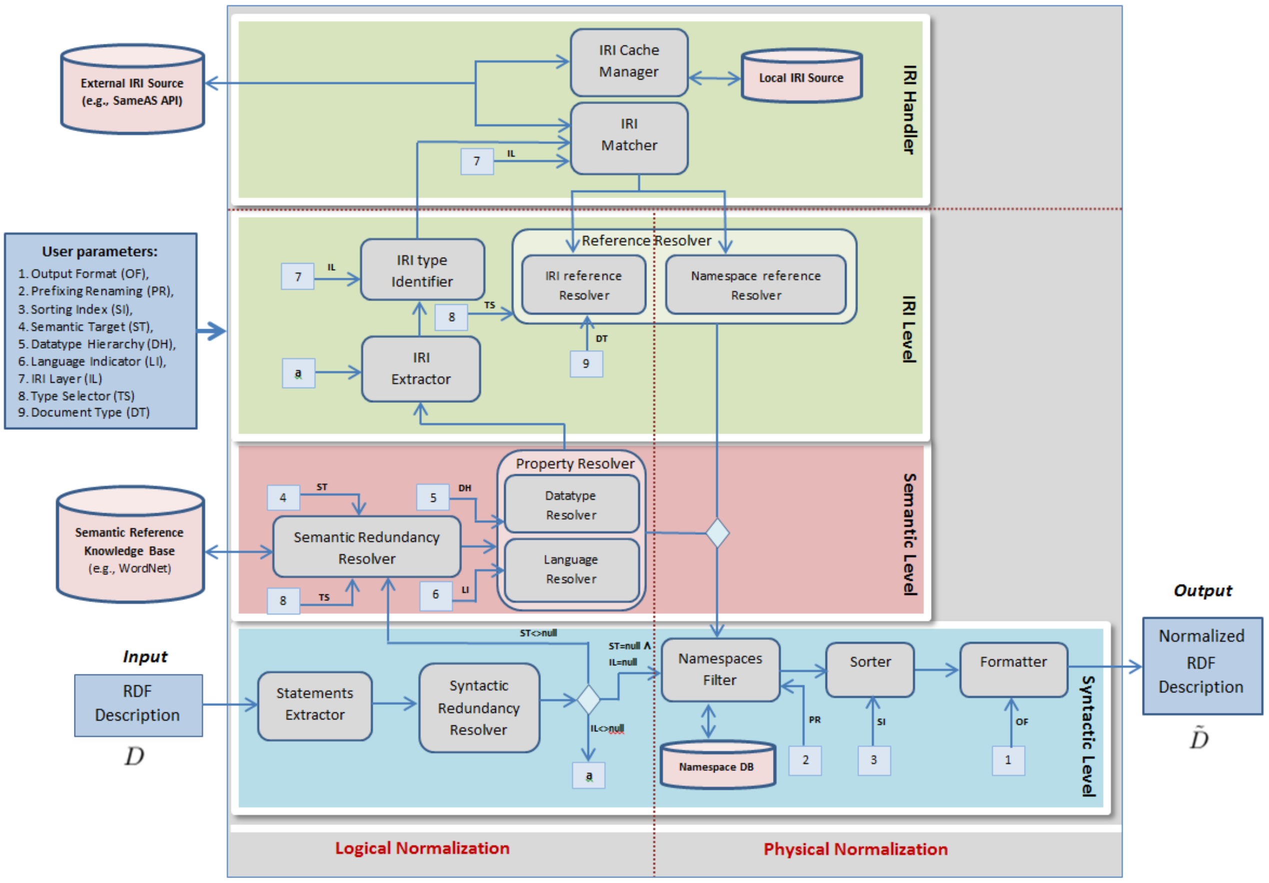


Figure 5.6: Overall architecture of our RDF normalization framework

Note that our framework is flexible such that the user can choose the kind of normalization to apply on an RDF description, based on her needs, i.e., performing: syntactic-only, syntactic-semantic, syntactic-IRI, or combined syntactic-semantic-IRI normalization.

Table 5.4: Summarized descriptions of sets used in our approach

#	Parameter	Allowed Values	Description
1	Output Format (OF)	v_1: flat, v_2: compact, v_3: full compact	Allows the user to choose between three types of formats defined in [THTC+15])
2	Prefixing Renaming (PR)	v_1: original, v_2: system (default), v_3: collective	Allows the user to choose between the original prefixes, the system renaming prefixes or the collective prefixes defined in [THTC+15]
3	Sorting Index (SI)	v_1: spo, v_2: pso, v_3: pos, v_4: osp, v_5: ops, v_6: sop (default)	Allows the user to choose between the six indexes for the elements of the statement (subject, predicate, object)
4	Semantic Target (ST)	v_1: bn, v_2: IRI, v_3: literal (default), v_4: null	Allows the user to choose the elements of the RDF Description to analyze semantically (blank nodes, IRIs, literals, or the combinations between them)
5	Datatype Hierarchy (DH)	v_1: user, v_2: standard (default)	Allows the user to choose the standard hierarchy (W3C) of the datatypes or insert the user hierarchy preference
6	Language Indicator (LI)	v_1: user option, v_2: null default: en	Allows the user to choose the language preference for the normalization (en, es, fr, etc.)
7	IRI Layer (IL)	v_1: network, v_2: data (default), v_3: null	Allows the user to choose the layer for the IRI analyzing (network or data)
8	Type Selector (TS)	v_1: generic, v_2: specific, v_3: short (default), v_4: long, v_4: threshold, v_5: another	Allows the user to choose the type of semantic and IRI evaluation based on the generic or specific information, the shortest or longest one or using a threshold, i.e., the user he will choose the parameter depending on the target application that he wants to use, e.g., storage (less expensive), loading, etc.
9	Document Type (DT)	v_1: RDF/XML, v_2: N3 (default), v_3: JSON-LD, v_4: another	Allows the user to choose the document type of semantic evaluation based on the generic or specific information, the shortest one or using a threshold

5.4.1 Semantic Level

This level is developed to resolve semantic ambiguities between literals and blank nodes in an RDF description. It only targets logical (RDF graph) normalization, allowing to eliminate all semantic-based blank node duplications and/or literal node duplications (discussed in Section 3.3.3), and obtain extended statements without semantic duplications (cf. Rules 1 and 2). It consists of two major components:

1. ***Semantic Redundancy Resolver (RR):*** it allows to eliminate all the redundancies (Rules 1 and 2) caused by the semantic ambiguities. The inputs of this component are the extended statements from the input RDF description and the type of selector (TS) parameter given by the user, and produces as output a transformed RDF description where semantically redundant statements have been eliminated (following user preferences expressed by TS).
2. ***Property Resolver (PR):*** activated when the statements contain datatypes and/or languages tags. It allows to eliminate the logical redundancies in statements with datatypes and language tags (Rules 2.1 until 2.6) caused by the semantic ambiguities. The inputs of this component are the extended statements obtained as output from the semantic redundancy resolver, and two optional parameters: Datatype Hierarchy (DT) and Language Indicator (LI), highlighting the user's preferences about working with a specific datatype hierarchy or language.

The pseudo-code of our *semantic redundancy resolver* is provided in Algorithm 3. Note that, the *property resolver* is activated in the algorithm 3, using the parameters DH and LI to select the semantic statement duplication.

5.4.2 IRI Level

This level is developed to resolve IRI identification and coreference discrepancies in an RDF description, to avoid IRI duplications. It targets both logical (RDF graph) and physical (RDF serialization) normalization.

6.2.1 Logical Normalization

It is the first step in IRI-level normalization, allowing to eliminate all the logical redundancies in the input RDF graph created by the IRI discrepancies (discussed in Section 3.4.4), and obtain extended statements without IRI duplications (cf. Rule 3). For this purpose, we divide logical normalization in three-components:

Algorithm 3 Semantic Redundancy Resolver

```
Input: $ST^+[]$ //List of Extended Statements of the RDF Description
    $TS, DH, LI$ //Parameters
    $KB$ //Knowledge base
Output: $ST^+[]$ //List of Extended Statements without semantic duplication
 1: N=$st^+$.length(); //Number of Statements in the list
 2: for i=1, i ≤ N, i++ do
 3:   for j=i+1, j ≤ N, j++ do
 4:     if $st^+[j]$.to = "bn" and $st^+[i]$.to = "bn" and $st^+[i]$.s = $st^+[j]$.s and $st^+[i]$.p = $st^+[j]$.p and ($st^+[i] \preceq st^+[j]$ or
        $st^+[j] \preceq st^+[i]$) then
 5:       $SemSelect(st^+_i, st^+_j, TS, DH, null)$
 6:       $remove((st^+_{sem} \cup O(st^+_{sem})), ST^+[])$; // remove blank node duplication - Rule 1
 7:     else
 8:       if $st^+$[i].to = "literal" and $st^+[i]$.s = $st^+[j]$.s and $st^+[i]$.p = $st^+[j]$.p and $Syn(st^+_i.o, st^+_j.o, KB)$ then
 9:         if DH<>null then
10:           $SemSelect(st^+_i, st^+_j, TS, DH, null)$
11:         if LI<>null then
12:           $SemSelect(st^+_i, st^+_j, TS, null, LI)$
13:         else
14:           $SemSelect(st^+_i, st^+_j, TS, null, null)$
15:         $remove(st^+_{sem}, ST^+[])$; // remove semantic statement duplication - Rule 2
16: return $ST^+[]$
```

1. ***IRI Extractor (IE):*** it extracts all the IRIs from the extended statements in an RDF Graph. The inputs of this component are the extended statements produced as output by property resolver, or the extended statements produced as output by the syntactic redundancy resolver (in the case the user wished to perform syntactic-IRI normalization only, i.e., without semantic normalization).

2. ***IRI type identifier (ITI):*** it implements the type identification step of each IRI (name recognition of the entity and type). The inputs of this component are all the IRIs with their respective extended statements from the RDF graph, and the IRI layer (IL) parameter that allows the user to choose between data or network IRI (cf. Section 5.2.2) evaluation.

3. ***IRI Reference Resolver (IRR):*** it allows to eliminate all the redundancies (Rule 3) caused by the IRI discrepancies in the RDF Graph. The inputs of this component are the sets of ambiguous extended statements (based on the type of IRI, identifying the same resource) and two parameters: Type Selector (TS) and the Document Type (DT). allowing to capture user preferences about the type of selection (e.g., short, generic, etc., according to the target application) and the type of serialization (format, e.g., RDF/XML, N3, etc.) when dealing with the "document presentation" IRI type. The pseudo-code of our *IRI reference resolver* is provided in Algorithm 4.

Algorithm 4 IRI Reference Resolver

```
Input: $ST^+[]$ //List of Extended Statements of the RDF Description
    $TS, IL$ //Parameter
Output: $ST^+[]$ //List of Extended Statements without IRI discrepancies
 1: N=$st^+$.length(); //Number of Statements in the list
 2: for i=1, i $\leq$ N, i++ do
 3:     for j=i+1, j $\leq$ N, j++ do
 4:         if $st^+$[i].to = “IRI” and $st^+[i]$.s = $st^+[j]$.s then
 5:             if $st^+[i]$.p = $st^+[j]$.p and $identify(st_i^+.o, IL) = identify(st_j^+.o, IL)$ then
 6:                 $StSelect(st_i^+, st_j^+, TS)$
 7:             else
 8:                 if $st^+[i]$.o = $st^+[j]$.o and $identify(st_i^+.p, IL) = identify(st_j^+.p, IL)$ then
 9:                     $StSelect(st_i^+, st_j^+, TS)$
10:             $remove(st_{amb}^+, ST^+[])$; // remove semantic statement duplication - Rule 3
11: return $ST^+[]$
```

6.2.2 Physical Normalization

It is the second step in IRI-level normalization, allowing to eliminate namespace duplications which can create IRI discrepancies (discussed in Section 3.4.5), and allows obtaining an RDF file without namespaces duplications (cf. Rule 4). It basically consists of one component: **Namespace Reference Resolver**, which accepts as input the sets of ambiguous namespaces in an RDF file, and eliminates duplicated ones (based on the type of IRI identifying the same vocabulary). We provide the pseudo-code of this component in Algorithm 5.

Algorithm 5 Namespace Reference Resolver

```
Input: $QN[]$ //List of Qnames of the RDF Description
    $TS, IL$ //Parameter
Output: $QN[]$ //List of Qnames without IRI discrepancies
 1: N=$st^+$.length(); //Number of namespaces in the list
 2: for i=1, i $\leq$ N, i++ do
 3:     for j=i+1, j $\leq$ N, j++ do
 4:         if $identify(qn_i.ns, IL) = identify(qn_j.ns, IL)$ then
 5:             $NamespaceSelect(qn_i, qn_j, TS)$
 6:             $remove(qn_{amb}, QN[])$ and $replace(qn_{amb}.ns, qn_{select}.ns, ST^+[])$; // remove namespace discrepancies - Rule
                4
 7: return $QN^+[]$
```

5.4.3 IRI Handler

This module is developed to solve the IRI reference problem (discussed in Section 3.4) which is required to detect and eliminate IRI duplications.

1. ***IRI Matcher***: it implements IRI disambiguation between IRIs that identify the same resource with the same type (which is used inside Function 11). The inputs of this component are all the IRIs with their respective types and extended statements, as well

as the IRI layer (IL) parameter. We provide the pseudo-code of this component in Algorithm 6.

2. ***IRI Cache Manager:*** it provides a storage for IRIs that generated after doing the matching between the IRIs using an External IRI Source (e.g. SameAs[1] API). It allows the matching component to request the information from the IRI cache manager and , in case it is not found, to the request it from an external (user designated) IRI source.

Algorithm 6 IRI Matcher

```
Input: $IRI_i, IRI_j$ //Iris to be match
    $IL$ //Parameter
    $Source$ //Parameter
Output: boolean //Boolean value
 1: $Matcher(IRI_i, IRI_j, Source)$
 2: return boolean
```

5.5 Summary

In this chapter, we propose an extension of our syntactic RDF normalization framework, i.e., R2NR (Chapter 4) by integrating two additional levels (semantic and IRI) and one component (IRI handler) detailed in Section 5.4.

To develop this extension in Section 5.1, we presented additional functions related to solving the semantic and IRI challenges, leading to the creation of our new normalization rules (Section 5.3). Our extended RDF normalization framework, titled R2NRE, allows to keep all the characteristic of our first approximation, adding two new specifications related to semantic and IRI normalization, where through functions an rules our method can eliminate all the logical redundancies and physical disparities in RDF descriptions related to semantic ambiguities and IRI discrepancies (identified in our motivation Chapter 3) and obtain a normalized RDF description.

In table 5.5, we compare our extended RDF normalization against state of the art approaches detailed in Section 5.2.4. Note that in the table 5.5: X means that the approach does not solve the problem, ✓* means that the approach solves partially the problem, and ✓ means that the approach solves the problem. As shown this table, our approach covers all the problems presented in Chapter 3, in contrast with existing techniques which neglect most of them.

In the next chapter, we present and analyze the results of our experimental evaluation and discuss what this means for the effectiveness and efficiency of our approach. Also, we compare our results against two main methods (HDT and JSON-LD).

[1]`http://sameas.org/`

Table 5.5: Relation between problems and RDF semantic and IRI Normalization approaches

Approaches	Problems										
	Syntactic Issues	Semantic Ambiguities					IRI Discrepancies				
		P8	P9	P9.1	P9.2	P10	P11	P12	P13	P14	P15
Krogstie et al. [KVS07]	X	X	X	X	X	X	X	✓*	X	✓*	✓*
Garcia-Castro et al. [GCGP10]	X	X	X	X	X	X	X	✓*	X	✓*	✓*
Kerzazi et al. [Kea07]	X	X	X	X	X	X	X	✓*	X	✓*	✓*
Kerzazi et al. [Kea08]	X	X	X	X	X	X	X	✓*	X	✓*	✓*
Jaffri et al. [JGM08b]	X	X	X	X	X	X	X	✓	X	X	X
Rusu et al. [RŠD+10]	X	X	X	X	X	X	X	✓	X	X	X
Shekarpour et al. [SNA12]	X	X	X	X	X	X	X	✓	X	X	X
Usbeck et al. [UNR+14]	X	X	X	X	X	X	X	✓	X	X	X
Our approach	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Chapter 6

Experimental Evaluation

"Simplicity is prerequisite for reliability."

— E. Dijkstra

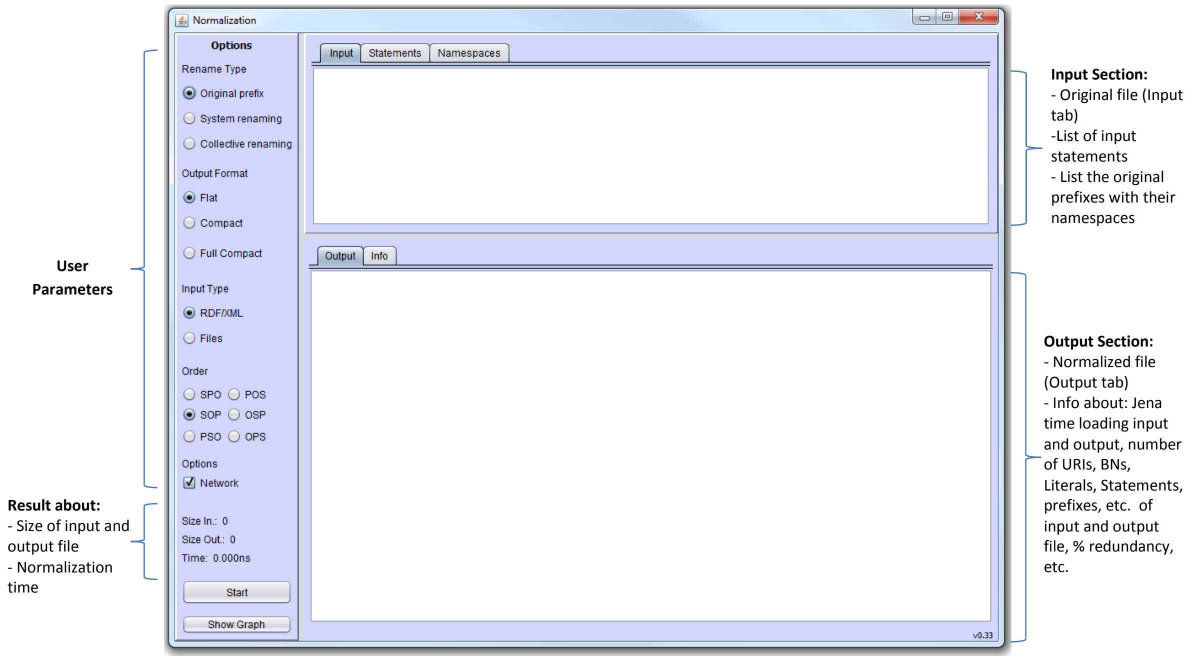
In the preceding two chapters we presented our RDF normalization approach. In this chapter, we describe our prototypes (Section 6.1), experimental metrics (Section 6.2), the experimental environment (See section 6.3), and the results of our experiments (Section 6.4). Experiments target the effectiveness, efficiency, jena loading time improvement, and storage improvement of RDF descriptions when performing normalization. Next, in Section 6.5, we also compare the quality of our approach with respect to JSON-LD and HDT methods. Finally, we wrap-us this chapter with a summary in Section 6.6.

6.1 Prototype

In order to validate our proposal, we have developed two versions of our prototype: online and desktop systems to test, evaluate, and validate our RDF Normalization framework. We call our prototypes ***RDF2NormRDF***. The desktop prototype system was developed using Java 7.0 (See Figure 6.1), whereas the online prototype system[1] was developed using PHP and Java as the RDF engine (See Figure 6.2). ***RDF2NormRDF*** was implemented following the R2NR architecture described in Figure 5.6. Hereunder, we describe the main components of the system:

- The ***logical normalization component***, accepts as input a (set of) RDF/XML file(s) and then parses the corresponding RDF descriptions, transforming them into extended

[1]Available at `http://rdfn.sigappfr.org/`


Normalization
Input
Statements
Namespaces
Output
Info
Start
Show Graph
v0.33
User Parameters
Result about:
- Size of input and output file
- Normalization time
Input Section:
- Original file (Input tab)
-List of input statements
- List the original prefixes with their namespaces
Output Section:
- Normalized file (Output tab)
- Info about: Jena time loading input and output, number of URIs, BNs, Literals, Statements, prefixes, etc. of input and output file, % redundancy, etc.

Options

Rename Type: [✓ Original prefix] System renaming Collective renaming

Output Format: [✓ Flat] Compact Full Compact

Input Type: [✓ RDF/XML] Files

Order: SPO POS [✓ SOP] OSP PSO OPS

Options: [✓ Network]

Size In.: 0
Size Out.: 0
Time: 0.000ns

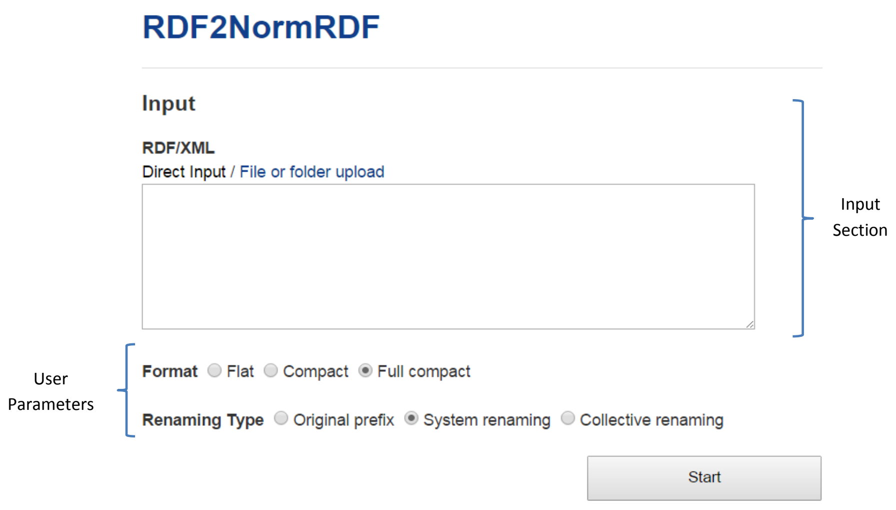
Figure 6.1: Desktop Prototype Interface.

statements while removing all logical redundancies, using Jena Libraries to parse the input file.

- The ***physical normalization component*** accepts as input: the output of the logical normalization component, in addition to prefix renaming and output format parameters, and then runs the latter through three consecutive processes: (a) *Namespaces Controller* (including the elimination of duplicate and unused namespaces), (b) *Sorting Processor* (producing statements in a specific order), and (c) *Formatting Processor* (producing the output normalized RDF description based on the output form chosen by the user).

Below, we detail the information of the tabs in the Output online interface (Figure 6.2):

- Output: the RDF/XML normalized document in accordance with the RDF/XML document input and the parameters of the format.
- Statements: The statements are printed with the subject, predicate and object values after the sorting process.
- Prefix: The prefixes and namespaces they represent after the normalization process, according to the input parameter about the prefix renaming.
- Info: in this part, we present the following information:
  - Normalization Time: It measures the total time of the process. The time computation is represented in nanoseconds.


RDF2NormRDF

Input

RDF/XML

Direct Input / File or folder upload

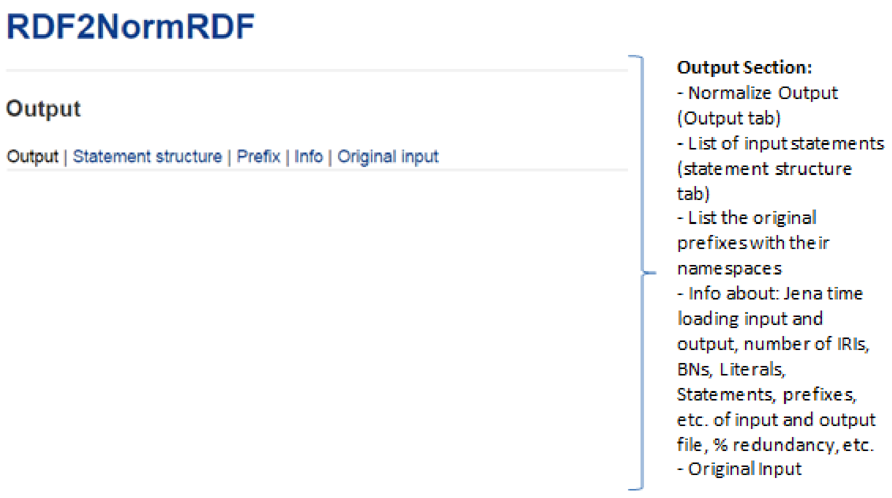
Input Section

User Parameters

Format ○ Flat ○ Compact [✓ Full compact]

Renaming Type ○ Original prefix [✓ System renaming] ○ Collective renaming

(a) Input


RDF2NormRDF

Output

Output | Statement structure | Prefix | Info | Original input

Output Section:
- Normalize Output (Output tab)
- List of input statements (statement structure tab)
- List the original prefixes with their namespaces
- Info about: Jena time loading input and output, number of IRIs, BNs, Literals, Statements, prefixes, etc. of input and output file, % redundancy, etc.
- Original Input

(b) Output

Figure 6.2: Online Prototype Interfaces

- Input Jena Time: the time to uploading in Jena the input file.
- Input size: The number of bytes in the input.
- Input IRI nodes: the number of IRIs in the input file.
- Input Blank nodes: the number of BNs in the input file.
- Input Literal nodes: the number of literals in the input file.
- Input Statements: the number of statements in the input file.
- Input Namespaces: the number of namespaces in the input file.
- Output Jena Time: the time to uploading in Jena the output file.
- Output size: The number of bytes in the output file.
- Output IRI nodes: the number of IRIs in the output file.
- Output Blank nodes: the number of BNs in the output file.
- Output Literal nodes: the number of literals in the output file.
- Output Statements: the number of statements in the output file.
- Output Namespaces: the number of namespaces in the output file.
- Results of % Redundancy Reduction.
- Results of % Disparity Reduction.
- Results of % Size Reduction.
- Results of % Reduction Jena Loading Time.

6.2 Experimental Metrics

We utilized three main criteria to evaluate the quality of our normalization approach: i) effectiveness, ii) efficiency, and iii) applicability.

6.2.1 Effectiveness (Ξ)

It is a boolean value that measures our normalization method (*logical normalization* process and *physical normalization* process) by assessing the resulting normalized RDF files w.r.t. the normalization goals and properties covered in Sections 4.3 and 4.4, such as:

$$\Xi(DS_i) = \Gamma(DS_i) \wedge \mathcal{P}(DS_i) \wedge \mathcal{K}(DS_i) \tag{6.1}$$

since:

$\Gamma(DS_i) = True$ if $\gamma_E(DS_i) = \gamma_I(DS_i)$

$$\gamma_E(DS_i) = \frac{\sum_{i=1}^{n} lr(D_i)}{n} \tag{6.2}$$

$$\gamma_I(DS_i) = \frac{\sum_{i=1}^{n} lr(D_i)}{n} \tag{6.3}$$

where:

- DS is the RDF dataset,
- n is the number of files in the dataset,
- Γ is the comparison between the average of logical redundancies in the input dataset group (DS_i) and the logical redundancies eliminated from the same dataset group for the method,
- lr is the percentage of logical redundancies in the file,
- γ_E and γ_I are the average of logical redundancies in the dataset group where E is the average of the eliminated redundancies and I is the average of the input of redundancies.

$\mathcal{P}(DS_i) = True$ if $\rho_E(DS_i) = \rho_I(DS_i)$

$$\rho_E(DS_i) = \frac{\sum_{i=1}^{n} pd(D_i)}{n} \tag{6.4}$$

$$\rho_I(DS_i) = \frac{\sum_{i=1}^{n} pd(D_i)}{n} \tag{6.5}$$

where:

- DS is the RDF dataset,
- n is the number of files in the dataset,
- $\mathcal{P}$ is the comparison between the average of physical disparities in the input dataset group and the physical disparities eliminated from the same dataset group for the method,
- pd is the percentage of physical disparities in the file,
- ρ_E and ρ_I are the averages of physical disparities in the dataset group where E is the average of the eliminated disparities and I is the average of the input of disparities.

$\mathcal{K}(DS_i) = True$ if:

$$\forall D_i \in DS_i / D_i \text{ is consistent}$$

where:

- DS is the RDF dataset,
- $\mathcal{K}$ is the evaluation of the consistency in the dataset group.

6.2.2 Efficiency (λ)

In addition to assessing the effectiveness of our method in producing normalized documents, we evaluated its time performance and its complexity, using the following measure:

$$\lambda(DS_i) = \frac{\sum_{i=1}^{n} pt(D_i)}{n} \tag{6.6}$$

where:

- n is the number of files in the dataset,
- pt is the average of processing time of the file.

6.2.3 Applicability

We also evaluated the impact of our normalization process in a practical application setting, evaluating: i) Jena (framework for building Semantic Web applications which has been used in several projects[1] and existing studies [GCGP10]) loading time (Π) and ii) RDF file storage space (Φ). Here, we used the following measures:

$$\Pi(DS_i) = \frac{\sum_{i=1}^{n} lt(D_i)}{n} \tag{6.7}$$

$$\Phi(DS_i) = \frac{\sum_{i=1}^{n} sr(D_i)}{n} \tag{6.8}$$

where:

- n is the number of files in the dataset,
- lt is the average of Jena loading time of the file,
- sr is the percentage of size reduction of the file.

[1] `https://jena.apache.org/about_jena/contributions.html`

6.3 Experimental Environment

6.3.1 Processing Context

The experiments have been done under the environment described in Table 6.1.

Table 6.1: Experimental Environment

CPU	Intel®Core(TM) i7 - 2600 + 3.4GHz
Memory	8.00GB
OS	MS Windows 7 Professional
Programming Environment	Sun JDK 1.7

6.3.2 Dataset Context

We conducted experiments on 11 datasets categorized in four groups:

- (***Group 1***): The first dataset group consists of two synthetic data-sets created based on the running examples covered in our study. This group was created to test the quality of our method when applied on user files with redundant information in blank nodes, literals, statements, namespaces and unused namespaces.

Syn_DS1 : It consists of 13 generated RDF/XML files with different characteristics manually tuned to highlight the behavior of our method, while varying the duplications of BNs, statements, literals, and namespaces. Files of the *Syn_DS1* dataset are heterogeneous w.r.t. file size, RDF output form (flat, compact, or full compact), as well as the number of IRIs, BNs, literals, and statements. Yet, the files are homogeneous w.r.t. the namespaces utilized (i.e., they contain only few namespace disparities, cf. Table 6.2).

Syn_DS2 : It consists of 114 generated RDF/XML files based on modifications in the running example with more statements, blank node duplications as well as duplicated namespace prefixes and unused namespaces on a bigger scale then *Syn_DS2* dataset. This dataset is heterogeneous w.r.t. the number of blank nodes, literals and statements (cf. Table 6.2).

- (***Group 2***): The second dataset group is based on real files from the online version of the Linked Open Data cloud. This group was created to test the quality of our method when applied on real files with few or without redundant information.

LGD : *LinkedGeoData*[1] dataset consists of 500 RDF/XML homogeneous files sharing similar features w.r.t. the number of nodes and the number of statements.

DBp : *DBpedia*[2] dataset consists of 355 RDF/XML heterogeneous files which mainly vary w.r.t. file size, the number of IRIs, and the number of statements.

WN : *WordNet*[3] dataset consists of 1087 RDF/XML heterogeneous files which vary w.r.t. the number of nodes, and the number of statements.

- (***Group 3***): The third group of datasets is generated by including additional random redundancies and disparities in two real datasets of the Group 2. This group was generated to test and measure the behavior of our method when handling typical files with redundant information.

LGD2 : It consists of 145 generated files of LinkedGeoData with statements and namespaces duplicated and a few unused namespaces.

DBp2 : It consists of 119 generated files of DBpedia with statements and namespaces duplicated and a few unused namespaces.

- (***Group 4***): The fourth dataset group was synthetically created based on the datasets of groups 1 and 3 (Syn_DS1, Syn_DS2, LGD2 and DBp2) after applying the*JSON-LD Normalization*[4] and *HDT technique*[5]. This group was used to test the efficiency of our approach in comparison with both *JSON-LD* and *HDT* normalization methods.

The main features of all datasets are summarized in Tables 6.2, 6.3, and 6.4. Note that in tables 6.3, and 6.4, the datasets after applying JSON-LD method have some logical redundancies but not present any physical disparities (due to the output format does not allow to have namespaces duplications) and datasets after applying HDT method do not present any logical redundancies or physical disparities (due to all blank node redundancies are represented as IRIs). A detailed evaluation about this results, we provide in Section 6.5, where we discuss about the behavior of other methods.

[1]`http://linkedgeodata.org/Datasets`
[2]http://wiki.dbpedia.org/Datasets
[3]http://wordnet-rdf.princeton.edu/
[4]`http://json-ld.org/playground/`
[5]http://www.rdfhdt.org/download/, HDT is a compact data structure and format for RDF. In order to compress the file, this technique reduces the verbosity, erasing some redundancies and assigning unique IDs to the elements (see Section 4.2.2).

Table 6.2: Features of files in each Dataset

Features	Group 1						Group 2									Group 3					
	Syn_DS1			Syn_DS2			LGD			DBp			WN			LGD2			DBp2		
	Max	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**
Size Kb	4.3	0.5	2.2	48.6	0.5	7.9	3.08	2.27	2.3	854.49	1.4	40.08	48.75	1.03	2.54	7.2	2.8	4.5	275.19	4.07	73.33
IRIs	17	2	6	47	2	6	11	8	9	4260	9	154	429	6	11	9	9	9	421	11	93
BNs	12	1	6	129	1	19	0			0			0			0			0		
Literals	26	3	14	320	3	50	25	6	6	218	0	53	133	2	17	40	7	20	851	1	160
Statements	63	5	30	784	5	116	33	14	14	7300	14	266	561	9	27	77	25	45	2062	33	454
BN Dup.	3	0	2	123	0	16	0			0			0			0			0		
Lit. Dup.	7	0	5	307	0	43	0			2	0	0.01	0			34	1	14	667	0	99
St Dup.	18	0	12	753	0	98	0			2	0	0.01	0			63	11	30	1314	18	275
Log. Red. %	69	0	32	98	0	64	0			1.27	0	0.004	0			77	29	57	75	37	52
Ns Dup.	4	0	2	126	0	14	0			0			0			7	1	3	14	3	9
Unused Ns	2	0	1	3	0	2	15	10	13	0			0			8	2	5	1	0	0.8
Phys. Dis. %	78	0	60	98	0	70	59	42	54	0			0			55	40	48	50	40	46

Table 6.3: Features of files in dataset Group 4 (after applying JSON-LD and HDT) based on Group 1

Features	Syn_DS1 - JSON-LD			Syn_DS1 - HDT			Syn_DS2 - JSON-LD			Syn_DS2 - HDT		
	Max	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**
Size Kb	4.64	0.56	2.23	5.58	0.64	2.67	37.17	0.56	6.03	47.65	0.64	7.57
IRIs	17	2	6	29	3	11	47	2	5	141	3	25
BNs	12	1	6	0			129	1	19	0		
Literals	25	3	13	25	3	13	259	3	40	259	3	40
Statements	54	5	24	54	5	24	400	5	66	400	5	66
BN Dup.	3	0	2	0			123	0	16	0		
Lit. Dup.	6	0	4	0			246	0	32	0		
St Dup.	9	0	6	0			369	0	49	0		
Log. Red. %	60	0	27	0			96	0	52	0		
Ns Dup.	0			0			0			0		
Unused Ns	0			0			0			0		
Phys. Dis. %	0			0			0			0		

Table 6.4: Features of files in dataset Group 4 (after applying JSON-LD and HDT) based on Group 3

Features	LGD2 - JSON-LD			LGD2 - HDT			DBp2 - JSON-LD			DBp2 - HDT		
	Max	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**
Size Kb	1.82	1.66	1.68	1.82	1.66	1.68	101.42	2.12	30.56	101.42	2.12	30.56
IRIs	9	9	9	9	9	9	421	11	93	421	11	93
BNs	0			0			0			0		
Literals	9	6	6	9	6	6	204	1	61	204	1	61
Statements	17	14	14	17	14	14	781	14	179	781	14	179
BN Dup.	0			0			0			0		
Lit. Dup.	0			0			0			0		
St Dup.	0			0			0			0		
Log. Red. %	0			0			0			0		
Ns Dup.	0			0			0			0		
Unused Ns	0			0			0			0		
Phys. Dis. %	0			0			0			0		

6.4 Experimental Results

6.4.1 Effectiveness (RDF Normalization Quality)

We verified, for both synthetic and real datasets, that our normalization goals were successfully reached, solving all identified problems (cf. Section 3), and verifying all normalization properties (cf. Section 4.4) as well.

In short, results clearly show in Table 6.5 that the effectiveness scores of our method in all the dataset groups are *True*, e.g. $\Xi(Syn\_DS1) = True$. In other words, the normalized RDF descriptions, that our approach produces, fulfill all predefined properties and goals in both real and synthetic datasets, i.e., eliminating the logical redundancies and physical disparities (only for the syntactic level[1]) while preserving the *consistency* of the files. Note that WN dataset does not have logical redundancies and physical disparities and after applying our method the dataset continue preserving the consistency. So, we prove that our method does not cause any variation in quality of original data.

In group 2, we have special cases: RDF files that they do not have redundancies and/or disparities, or contain few redundancies as ***DBp***, however our approach does not cause any negative impact in the datasets and thus preserves the consistency of such files.

Table 6.5: Goals and properties achieved in the Datasets

Goals/Properties	Group 1		Group 2			Group 3	
	Syn_DS1	**Syn_DS2**	**LGD**	**DBp**	**WN**	**LGD2**	**DBp2**
Solving logical redundancies (%input=%erase)	32%	64%	0%	0.004%	0%	57%	52%
Solving physical disparities (%input=%erase)	60%	70%	54%	0%	0%	48%	46%
Preserving completeness	True	True	True	True	True	True	True
Preserving minimality	True	True	True	True	True	True	True
Preserving compliance	True	True	True	True	True	True	True
Preserving consistency	True	True	True	True	True	True	True

6.4.2 Efficiency (Time Performance)

In fact, the complexity of our method comes down to worst case O(N^2) time where N represents the number of RDF statements in the target RDF description D, since our main normalization

[1]The semantic and IRI levels will be evaluate in future works

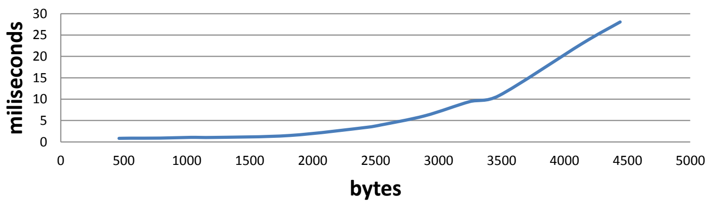


Figure 6.3: Average normalization time in the Syn_DS1 dataset which RDF descriptions contain a considerable amount of logical redundancies and physical disparities.

process (cf. Algorithm 1 - Redundancy Controller) needs to process each statement in D against all others in D in order to identify and then solve redundancies/disparities, given that output (normalized) statement re-ordering and serialization (cf. Algorithm 2 - Statements Sorter) requires average linear time.

Note that our overall normalization process can reach average linear complexity levels, i.e., best case O(N), when processing RDF descriptions with no or very few redundancies/disparities: where the normalization process would simply come down to sorting statements (cf. Algorithm 2).

Experimentally, we evaluate our method's time performance with each group[1]:

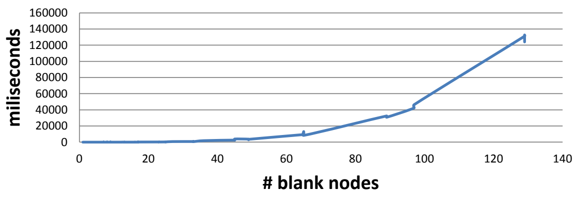
- ***Group 1***: First in the ***Syn_DS1*** dataset, we verify our method's polynomial time (almost linear) dependency on the size of the RDF file, containing blank nodes, logical redundancies, and physical disparities (see Figure 6.3).

  Second in ***Syn_DS2***, we verify our method's polynomial time (almost linear) dependency on the amount of blank nodes of the RDF file, containing logical redundancies and physical disparities (see Figure 6.4.a). Meanwhile, time dependency on RDF file size becomes relatively trivial when the amount of the blank nodes is significant in the files as shown in Figure 6.4.b[2].

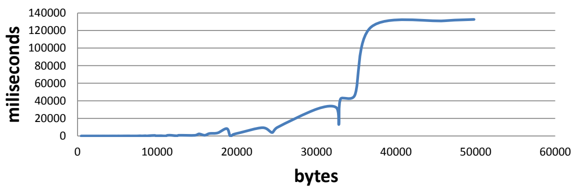
- ***Group 2***: Results in Figure 6.5.a show that processing time is almost linear w.r.t. file size when the amount of logical redundancies and/or physical disparities in the file is limited such as with ***LGD*** (see Figure 6.5.a) and when the amount of IRIs and statements is homogeneous in the dataset. However, processing time becomes polynomial when the files have a considerable amount of IRIs and statements without logical redundancies and/or physical disparities such as in the ***DBp*** and ***WN*** datasets (see Figure 6.5.b and c)

[1]All the tests related to time processing were executed 10 times, and for the evaluation we used an average value of the 10 executions.

[2]Note that the variation in the behavior of Figure 6.4.b is because a file of the dataset (Syn_DS2) contains a shorter number of BNs w.r.t. the others files.

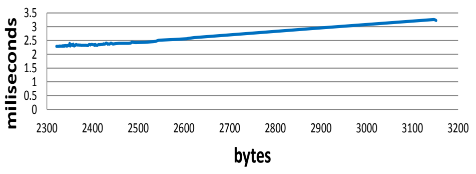


(a) Syn_DS2 normalization time evaluated w.r.t. the number of BNs.

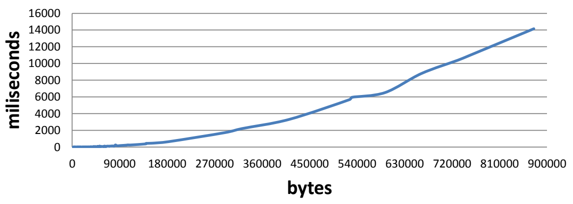


(b) Syn_DS2 normalization time evaluated w.r.t. file size.

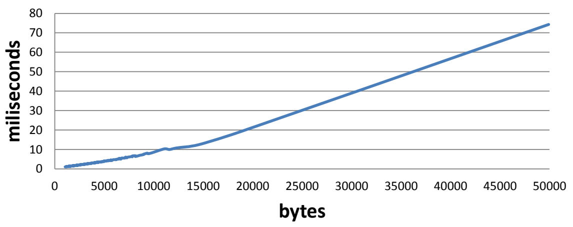
Figure 6.4: Average normalization time in Syn_DS2 dataset.



(a) LGD contains only physical disparities without logical redundancies. It presents a linear time.



(b) DBp without physical disparities and 0.004% of logical redundancies. It presents an average linear time.



(c) WN without logical redundancies and physical disparities. It presents an average linear time.

Figure 6.5: Average normalization time of the datasets in Group 2.

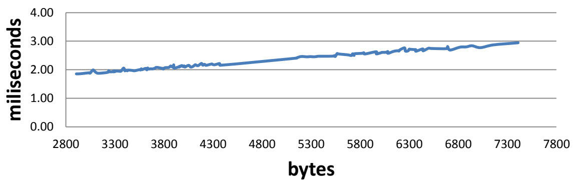
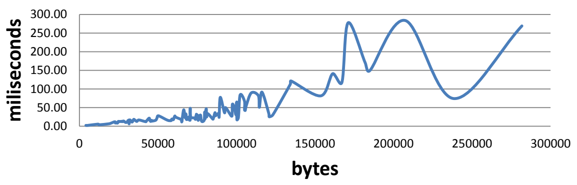
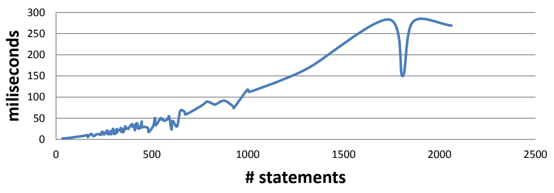


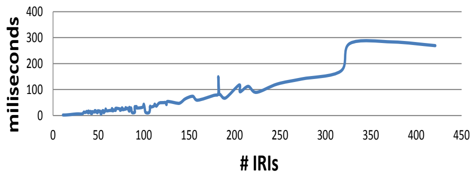
Figure 6.6: Average normalization time of the LGD2 dataset, with limited logical redundancies and physical disparities. It presents a linear time.



(a) DBp2 evaluated w.r.t. the size.



(b) DBp2 evaluated w.r.t. the number of statements



(c) DBp2 evaluated w.r.t. the number of IRIs

Figure 6.7: Average normalization time of the DBp2 dataset (containing a considerable amount of logical redundancies and physical disparities, cf. Table 6.2).

- ***Group 3***: Results in Figure 6.6 show that processing time is almost linear w.r.t. file size when the files have logical redundancies and/or physical disparities, and the amount of IRIs and statements is homogeneous in the dataset, such as with ***LGD2***. On the other hand, ***DBp2*** has a considerable amount of IRIs and statements and the files are not homogeneous (cf. Table 6.2), therefore, the results vary. As we show in Figure 6.7, time dependency on RDF file size becomes relatively trivial when the amount of IRIs and statements is bigger in the dataset, and when the files are not homogeneous w.r.t these variables. For example, we can see variations in Figure 6.7.b because the amount of IRIs is decreasing in some files.

Morover, we evaluated the impact of the sorting indexes order (cf. Section 4.5.2.2) in the Normalization Time. Results in Figure 6.8 show that index SOP (subject-object-predicate) highlights the best time performance, whereas SPO underlines the worst time performance among all six indices.

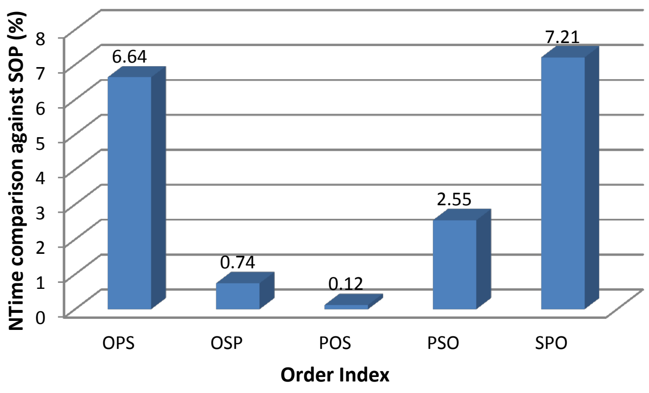
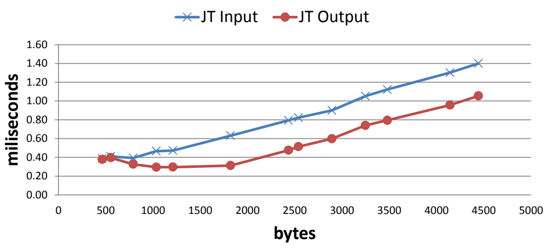
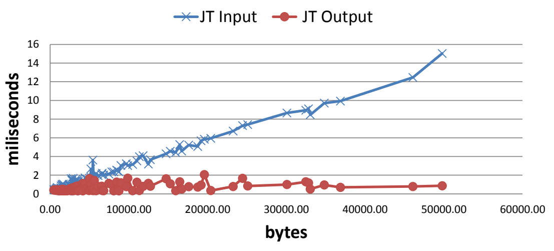


Figure 6.8: Order comparison ratio against normalization time of the SOP order.



(a) Syn_DS1



(b) Syn_DS2

Figure 6.9: Average Jena loading time of the datasets in Group 1.

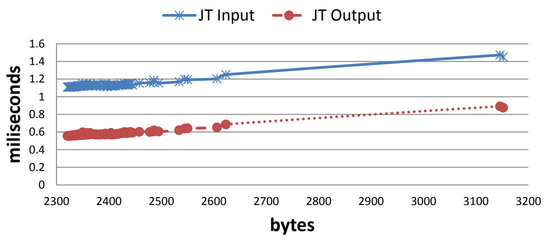
6.4.3 Applicability

A. **Jena loading time**:

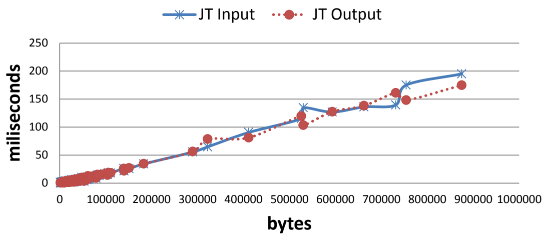
Results, with respect to all three groups, indicate that our approach improved the files' loading time in comparison with the original loading time of the datasets.

- ***Group 1***: Results shown in Figure 6.9 concur with those shown in the previous subsection, such that loading time becomes polynomial when normalizing files of the ***Syn_DS1*** and ***Syn_DS2*** datasets having logical redundancies and physical disparities, where the increase of loading time varies depending on the amount of blank nodes (as shown in Figure 6.9.b).

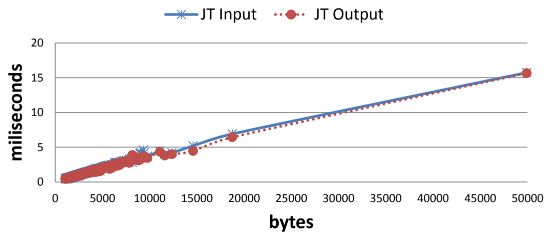
- ***Group 2***: Results shown in Figure 6.10.a concur with those shown in the previous subsection, such that loading time is almost linear in the size of files in the ***LGD*** dataset since they contain very few redundancies/disparities and are more or less



(a) LinkedGeoData - Dataset



(b) DBpedia - Dataset



(c) WordNet - Dataset

Figure 6.10: Average Jena loading time of the datasets in Group 2.

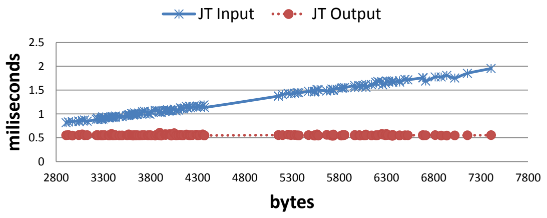
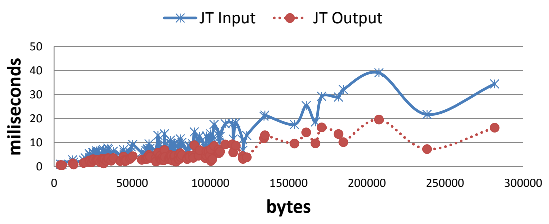


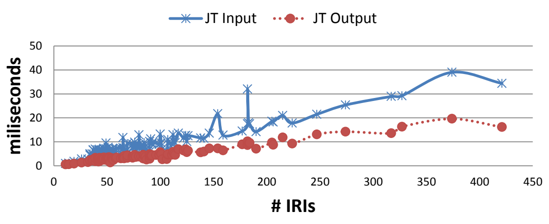
Figure 6.11: Average Jena loading time of RDF files in the LGD2 dataset.

homogeneous w.r.t. the amount of IRIs and statements. In contrast, loading time in Figures 6.10.b and c is polynomial w.r.t. the amount of IRIs and statements in ***DBp*** and ***WN*** datasets.

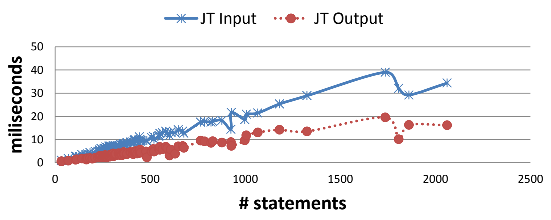
- ***Group 3***: In Figure 6.11, results shown that loading time is linear when we remove all the logical redundancies and physical disparities and the dataset is homogeneous w.r.t. the amount of IRIs and statements as ***LGD2***. In Figure 6.12, one can conclude that time dependency on file size becomes trivial when the amount of IRIs and statements increases as in ***DBp2***. For example, in Figure 6.12.b shows that loading time varies with file size because the files are not homogeneous, i.e., the amount of statements increases but the amount of IRIs decreases in some files.



(a) DBp2 evaluated by the size



(b) DBp2 evaluated by the number of IRIs



(c) DBp2 evaluated by the number of statements

Figure 6.12: Average Jena loading time of RDF files in the DBp2 dataset.

More significant, results in Figure 6.13 show that even when applied on homogeneous files (e.g., WordNet) without or with a limited amount of logical redundancies and physical disparities (e.g., DBpedia), our method continues improving the Jena loading time due to the format of our outputs (flat, compact and full compact). After analyzing the results of each data-set, one can conclude that Jena loading time improves (i.e., is reduced) as a direct result of normalizing the data-sets, taking into account RDF file size, as well as the amount of statements, IRIs, and blank nodes in the RDF graph (cf. Figure 6.13).

In addition, we evaluated the impact of the sorting indexes order (cf. Section 4.5.2.2) in Jena loading time. Results in Figure 6.14 show that index PSO (predicate-subject-object) highlights the best time performance, whereas SPO underlines the worst time performance among all six indices.

B. **Storage**:

Results, w.r.t. all three groups, indicate that our method reduces RDF file output size in comparison with the original size of the files. Based on our output formats: *flat*, *compact* and *full compact*, we show that our method is adaptable to different target applications. For instance, if the application requires improving the compression ratio of the file, the *full compact* format would more suitable than other formats.

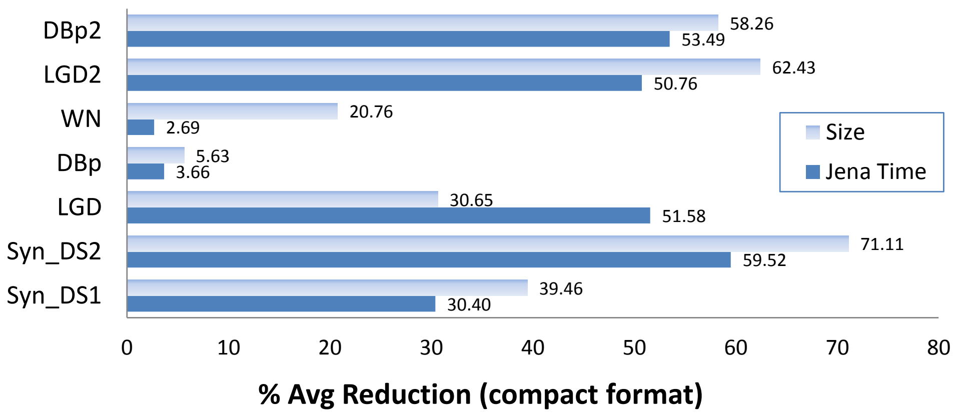


Figure 6.13: Average size reduction results w.r.t. the compact format.

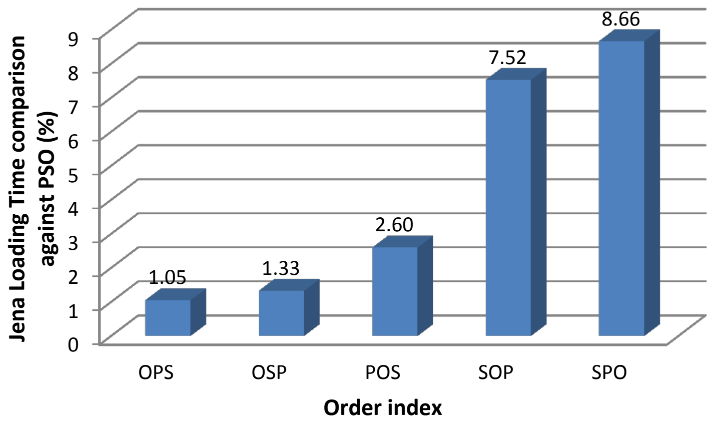


Figure 6.14: Order comparison ratio of the PSO sorting index order against Jena loading time.

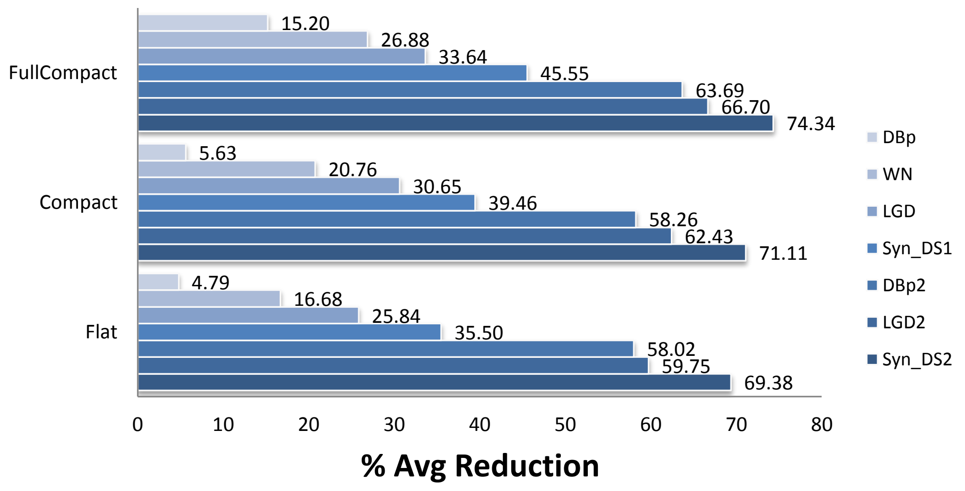


Figure 6.15: Average size reduction of the Datasets w.r.t the output format

We also analyzed the full compact format against the behavior of the other formats. Results in Figure 6.15 show that it has higher ratio in the results. For example, it produces:

- An average of 74.34% reduction in size of ***Syn_DS2*** (dataset with a considerable amount of logical redundancies and physical disparities) in comparison with a 69.38% reduction using the flat format,
- An average of 15.2% reduction in size of ***DBp*** (dataset without physical disparities and with minimal (0.004%) logical redundancies) in comparison with a 4.79% reduction using the flat format, and
- An average of 26.88% reduction in size of ***WN*** (dataset without physical disparities and logical redundancies) in comparison with a 16.68% reduction using the flat format.

6.5 Comparison with existing Methods

We also evaluated the quality of our approach in comparison with alternative methods, namely the JSON-LD normalization approach [SL13] and the HDT (RDF compression) method [Fea13], using dataset Group 4 (see Tables 6.3 and 6.4).

It is worthy to note that we only compared our approach with JSON-LD and HDT methods, since most other methods are theoretical, or incomplete, or do not provide accessible prototypes.

6.5.1 Effectiveness (RDF Normalization Quality)

We compared the effectiveness of our approach in comparison to alternative methods by assessing the nature and properties of resulting normalized RDF files w.r.t. our set normalization problems and goals (cf. Sections 3 and, 4.3).

i) First, results in Tables 6.6 and 6.7 show that our method produces normalized RDF files that fulfill all our normalization properties and goals in comparison with JSON-LD and HDT methods which miss several logical and physical redundancies/disparities.

ii) Second by comparing the original input in ***Syn_DS1***, ***Syn_DS2***, ***LGD2*** and ***DBp2*** w.r.t. the outputs of JSON-LD, HDT and our method (see Tables 6.8 and 6.9), we further verified that the goals and properties are not successfully reached for both approaches:

- ***JSON-LD method***: it preserves some redundancies and disparities, i.e., in Table 6.8, we show that JSON-LD removes only 5% over a 32% average of logical redundancies in the ***Syn_DS1*** dataset, and 12% over a 64% average of logical redundancies in the ***Syn_DS2*** dataset. However, it removes all logical redundancies and physical disparities from the ***LGD2*** and ***DBp2*** datasets (all 57% and 52% of logical redundancies and 48% and 46% of physical disparities were removed). A careful inspection of JSON-LD shows that it preserves blank node duplication and certain literal duplications, which explains the results obtained with ***LGD2*** and ***DBp2*** (since it does not contain any blank nodes). As a result, the JSON-LD approach does not satisfy the *minimality* and *consistency* properties (cf. Section 4.4).

- ***HDT method***: results show, at first glance, that it successfully eliminates logical redundancies and physical disparities (see Table 6.9). Nonetheless, a closer look at the results revealed that the HDT technique actually preserves blank node redundancies by assigning them different identifiers and/or representing them as IRIs. Hence, the HDT method actually keeps logical redundancies and physical disparities pertaining to blank node related statement duplications, and does not consequently satisfy the *completeness*, *minimality* and *consistency* properties.

Hence, overall results indicate that our method yields improved effectiveness (i.e., normalization quality) in comparison with current alternative methods.

Table 6.6: Goals and properties achieved in the Dataset Group 4 based on Group 1

Goals/Properties	Group 4 based on Group 1			
	Syn_DS1 - JSON-LD	**Syn_DS1 - HDT**	**Syn_DS2 - JSON-LD**	**Syn_DS2 - HDT**
Solving logical redundancies (%input=%erase)	27%	0%	52%	0%
Solving physical disparities (%input=%erase)	0%	0%	0%	0%
Preserving completeness	True	True	True	True
Preserving minimality	True	True	True	True
Preserving compliance	True	True	True	True
Preserving consistency	True	True	True	True

Table 6.7: Goals and properties achieved in the Dataset Group 4 based on Group 3

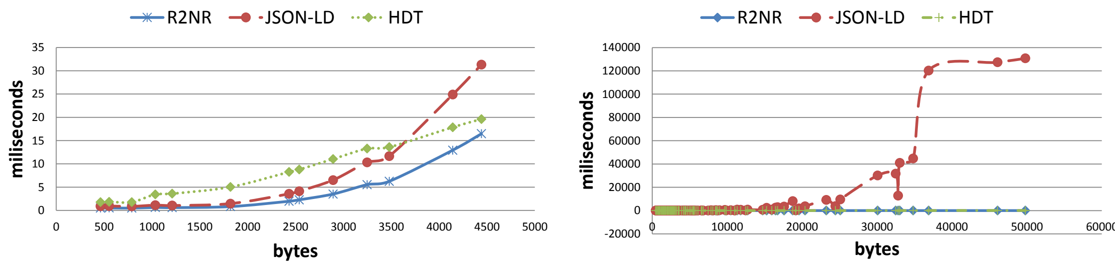
Goals/Properties	Group 4 based on Group 3			
	LGD2 - JSON-LD	**LGD2 - HDT**	**DBp2 - JSON-LD**	**DBp2 - HDT**
Solving logical redundancies (%input=%erase)	0%	0%	0%	0%
Solving physical disparities (%input=%erase)	0.17%	0.17%	0%	0%
Preserving completeness	True	True	True	True
Preserving minimality	True	True	True	True
Preserving compliance	True	True	True	True
Preserving consistency	True	True	True	True

Table 6.8: Goals and properties achieved in the Dataset Group 1 after applying normalization processes

Goals/ Properties	**Group 1**					
	Syn_DS1 - JSON-LD	**Syn_DS1 - HDT**	**Syn_DS1 - R2NR**	**Syn_DS2 - JSON-LD**	**Syn_DS2 - HDT**	**Syn_DS2 - R2NR**
Solving logical redundancies	5%	32%	32%	12%	64%	64%
Solving physical disparities	60%	60%	60%	70%	70%	70%
Preserving completeness	True	False	True	True	False	True
Preserving correctness	True	True	True	True	True	True
Preserving minimality	False	False	True	False	False	True
Preserving consistency	False	False	True	False	False	True

Table 6.9: Goals and properties achieved in the Datasets of Group 3 after applying normalization processes

Goals/ Properties	**Group 3**					
	LGD2 - JSON-LD	**LGD2 - HDT**	**LGD2 - R2NR**	**DBp2 - JSON-LD**	**DBp2 - HDT**	**DBp2 - R2NR**
Solving logical redundancies	57%	57%	57%	52%	52%	52%
Solving physical disparities	48%	48%	48%	46%	46%	46%
Preserving completeness	True	False	True	True	False	True
Preserving correctness	True	True	True	True	True	True
Preserving minimality	True	True	True	True	True	True
Preserving consistency	True	True	True	True	True	True



(a) Syn_DS1

(b) Syn_DS2

Figure 6.16: Comparison of the Average normalization time with JSON, HDT and R2NR.

6.5.2 Efficiency (Time Performance)

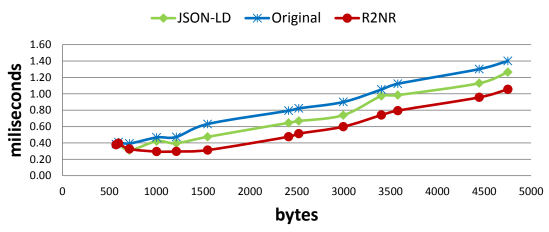
We evaluated the time performance of JSON-LD and HDT methods our approach's normalization time. Results in Figure 6.16.a show that our method also performs better than its alternatives (our method's shows a 46.77% of average reduction in normalization time compared to JSON-LD, and 52.18% of average reduction in time compared to HDT) because it erases all the logical and physical disparities in the dataset, including blank node duplications. Results in Figure 6.16.b show that our method is performs better than JSON-LD method (highlighting a 99.93% of average reduction in normalization time)[1]. As one can observe in Figure 6.16.b the temporal behavior of our method is similar to HDT method because the latter converts the blank nodes into IRIs. This conversion reduces processing time but preserves redundancies. Our method has 3.298 milliseconds of average in processing time while the HDT method has 7.34 miliseconds of average in processing time. Therefore, our method also shows better results than HDT.

Note that reducing redundancies means reducing document size, erasing duplication in blank nodes, literal, statements, which consequently reduces processing and loading time to produce the normalized output files.

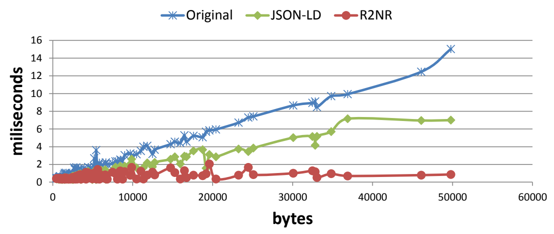
6.5.3 Applicability

We also tested the efficiency and the impact of our method in Jena loading time and RDF file storage size.

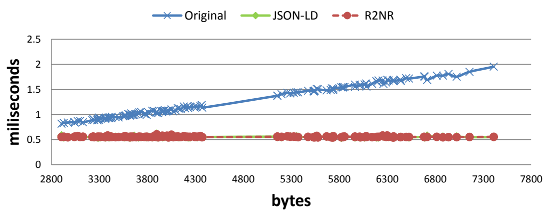
[1]Note that the normalization time of Syn_DS2 using JSON-LD method keeps the variation in the behavior of Figure 6.16.b because JSON-LD preserves all the BNs duplication. As we show before, Syn_DS2 dataset has a file that contains a shorter number of BNs causing a faster normalization time w.r.t. the others files.



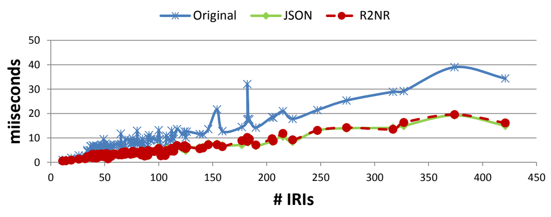
(a) Syn_DS1



(b) Syn_DS2



(c) LGD2



(d) DBp2

Figure 6.17: Average Jena loading time comparison with JSON-LD.

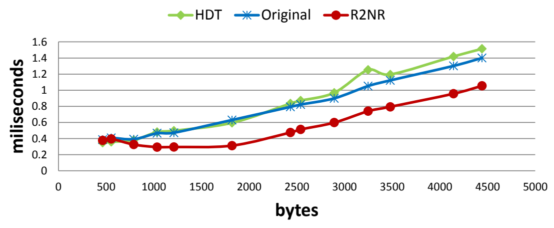
A. **Jena loading time**:

- ***JSON-LD method***: On one hand, Figures 6.17.a, 6.17.b, 6.17.c, and 6.17.d depict Jena loading time in comparison with our approach's time and that of *JSON-LD*'s approach. Results demonstrate that our method executes faster than JSON-LD's. Note that redundancy reduction using of *JSON-LD* amounts to 5% on average file size in the Syn_DS1, while our method reaches an average 27% size reduction ratio, which explains the reduction in loading time.

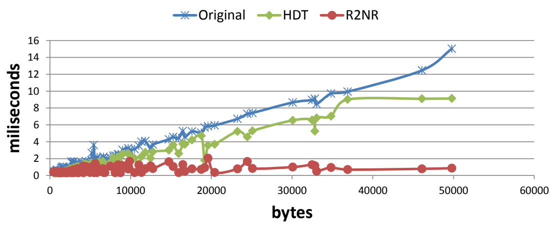
- ***HDT method***: On the other hand, results in Figures 6.18.a, 6.18.b, 6.18.c, and 6.18.d show that our method remains also faster than the HDT method. In fact, as shown in Table 6.8 and 6.9, the datasets generated by HDT do not have redundancies and present some disparities, yet contain a larger number of IRIs with no (zero) BNs. This confirms that HDT is transforming BNs into IRIs, which shows that RDF compression does not always guarantee normalization. Note that we are currently investigating this issue in more detail in a dedicated experimental study.

B. **Storage**:

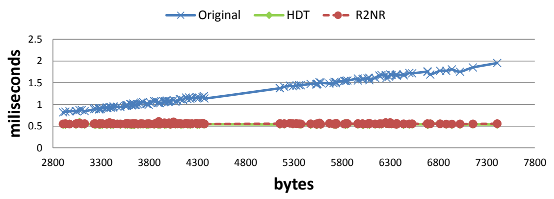
Neither JSON-LD nor HDT methods provide parameters to customize output format requirements as we do. They work with their predefined outputs, i.e., the JSON-LD produces files serialized as N-triples, and HDT produces Bitmap Triples (BT), in comparison



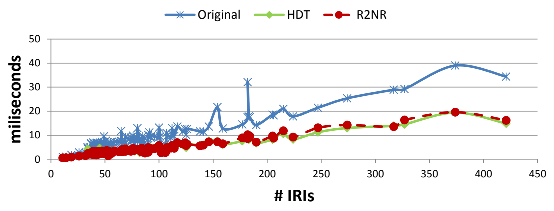
(a) Syn_DS1



(b) Syn_DS2



(c) LGD2



(d) DBp2

Figure 6.18: Average Jena loading time comparison with HDT.

with our method which handles the standard formats with different configurations (i.e., flat, compact, and full compact, cf. Section 4.5.2.3) and thus allows developing different outputs w.r.t the target application. Empirically, results in Figure 6.19 show that our normalization approach improves the size of the RDF files in all formats of the datasets processed by JSON-LD and HDT methods.

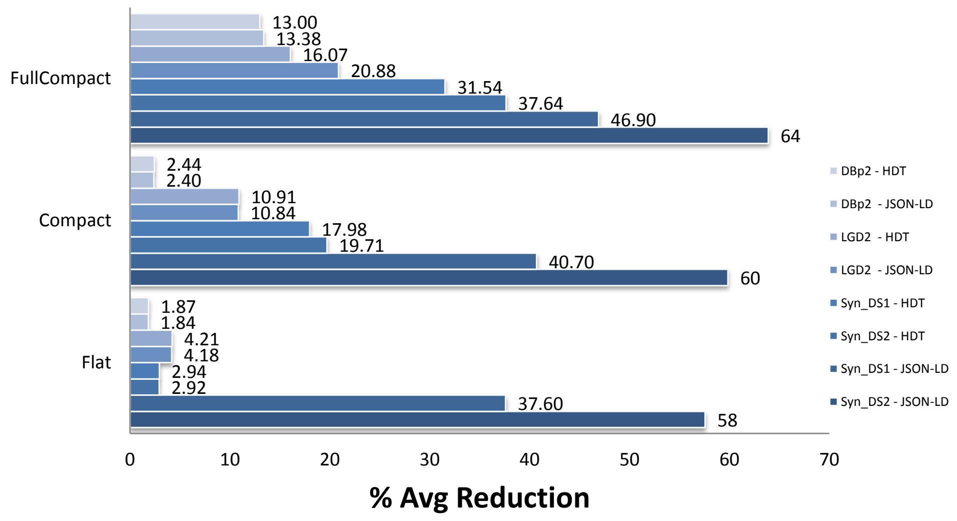


Figure 6.19: Average size reduction in dataset Group 4 w.r.t. the output format

6.6 Summary

As we have seen in this chapter, our RDF normalization approach has produced rather mixed results with a large degree of variation over different datasets with respect to Jena loading time, normalization time and storage. These variations correspond to the different natures of each dataset, i.e., some are homogeneous datasets and another heterogeneous datasets. The heterogeneity of the dataset depends of several variables as number of IRIs, number of statements, number of blank nodes, redundancy, etc.

Despite these variations, results are promising and clearly show that our approach can work for cleaning the RDF descriptions. Therefore, we can say that we have successfully fulfilled all the challenges presented in Sections 3.1 and 3.2. We have demonstrated that our RDF normalization approach can be applied to an RDF description and provide an RDF normalized description successfully.

In terms of fulfillment of our properties, we saw a score well on effectiveness, efficiency, jena loading time improvement and storage improvement for the output files with respect to the input files of each dataset. What this served to demonstrate is that our approach works for all our datasets.

At the same time with all the datasets, we also showed that our approach can lead to significant improvements with respect to other methods as JSON-LD and HDT. In other words, results showed that our approach solves all the logical redundancies and physical disparities detailed in Section 3 and also gives improvements in the RDF/XML formats to reduce the storage, all of which are not considered in existing methods.

Another interesting point that has been raised by these experiments is the impact of our parameters that give to the user more flexibility to handle his/her requirements following a specific target application. For example, if the target application is based on compression, the user can choose full compact format with system renaming to obtain a compressed RDF normalized description.

In the near future, we plan to test the impact and effect of applying our normalization approach on native RDF database systems, using a public benchmark such as LUBM[1] to evaluate database-related parameters such as: indexing time, storage space, query evaluation time, among others, in order to further evaluate and validate our solution.

[1] `http://swat.cse.lehigh.edu/projects/lubm/`

Chapter 7

Conclusions and Future Works

"We can't blame the technology when we make mistakes."
— Tim Berners-Lee

In this thesis, we have proposed and evaluated an RDF normalization framework, called R2RN, that eliminates logical redundancies and physical disparities from RDF descriptions in order to provide a normalized RDF descriptions. We have proposed a formal mathematical model with rules, functions, operators, properties, and proofs which allowed us to validate our proposal. We also conducted a thorough analysis of state of the art methods, highlighting the properties and limitations toward RDF normalization. In this chapter, we present our conclusions on our work and discuss some of the wider issues around the challenges. Finally, we present a number of ideas for future works that could be undertaken to extend our work and further our aim to other topics in the Semantic Web as the Web Ontology Language - OWL.

7.1 Recap

Chapter 2 covered the background review, in which we focused on basic notions about the World Wide Web - WWW, International Resource Identifier - IRI, Resource Description Framework - RDF, and the Web of Linked Data - LD. We also discussed the principles of the Web related to the RDF and Linked Data and the relationships between them to the RDF data on the Web. Then, we investigated the evolution of the Web of Linked Data and its impact on the number of projects and providers using RDF to share and connect the information.

Chapter 3 described different use cases to illustrate the problems motivating our work, categorized in four levels: logical redundancies, physical disparities, semantic ambiguities, and IRI discrepancies. Note that, the two first levels are the bases of the normalization challenges, whereas the last two levels can produce more logical redundancies and physical disparities for the normalization challenges.

Chapter 4 developed our RDF normalization framework called *R2NR*, as a means to transform RDF descriptions into normalized representations. Our approach allows to:

1. Preserve all the information in RDF descriptions,
2. Eliminate all the logical redundancies and physical disparities in the output RDF description,
3. Establish a unique specification of the statements in the RDF output description,
4. Formalize the normalization process,
5. Consider user parameters to handle the application requirements and adapt RDF output formats accordingly.

To our knowledge, this is the first attempt to study and integrate RDF normalization in two aspects: logical redundancies and physical disparities. Understanding that the presence of logical redundancies in RDF descriptions would have a negative impact on the processing of RDF information, as well as on the development/deployment of RDF databases and related applications (including storage, querying, similarity-based matching, and versioning, among others), our theoretical proposal showed that our approach helps alleviate the problem by eliminating all identified redundancies which were motivated in our study.

Chapter 5 presented our proposed extension of the RDF Syntactic Normalization in Chapter 4 where it is possible to normalize RDF descriptions taking into account: semantic ambiguities and IRI discrepancies. The semantic level allows to:

1. Eliminate all the logical redundancies generated for semantic problems as synonymy and ambiguity,
2. Preserve the semantic meaning in RDF descriptions,
3. Extend the R2RN framework with the semantic rules and functions.
4. Extend user parameters to handle the semantic application requirements.

The IRI level allows to:

1. Eliminate all the logical redundancies and physical disparities generated for IRI problems as coreference,
2. Preserve all the information in RDF description without the redundancies,
3. Extend the R2RN framework with the IRI rules and functions.
4. Extend user parameters to handle the IRI discrepancies.

With these two additional levels, our approach cover all the challenges that we identified in Chapter 3. Due to solving all the problems, our approach allows to have a complete RDF normalized description based on erasing in each level the logical redundancies and physical disparities.

Chapter 6 describes our prototype implementation for the R2NR framework, experimental metrics used to assess the quality and effectiveness of our approach, the experimental environment with our 11 datasets (synthetic and real), and our experimental results. This evaluation has revealed the need for a normalization process that can clean RDF descriptions of all the logical redundancies and physical disparities through the syntactic level and the impact of this process on the reduction of the storage and loading time.

Extensive experimental results confirm the positive impact of our normalization approach in terms of i) effectiveness, ii) efficiency, iii) applicability, and vi) storage space, in comparison with two of its most recent alternatives, confirming that the presence of logical redundancies and physical disparities in RDF descriptions would have a negative impact on the processing of RDF information, as well as on RDF databases and target applications.

Our experimental evaluation reinforces the theoretical validation presented in Chapter 4 to solve the RDF normalization problem by eliminating all identified redundancies and disparities which were motivated in our study.

7.2 Future Works

In this section, we discuss a number of possible avenues for future works that would advance our own research motivated by issues that our study has raised. Possible future directions include improvements to the RDF Normalization approach, Extended Statement Recommendation Format, Guidelines for Generating Normalized RDF descriptions, Ontology Normalization, RDF normalization for educational purposes, and Pre-processing phase for Web applications. These topics are now briefly described:

7.2.1 Improvements to the RDF Normalization Approach

In chapters 4 and 5, we have shown that RDF Normalization can be undertaken at three levels: syntactic, semantic, and IRI, and also takes into account two aspects: logical redundancies and physical disparities. These levels have specific topics that we can improve:

1. Syntactic level:

   (a) Adding the reification topic for evaluation.

(b) Generating more RDF serializations in the prototype and evaluating the results with respect to the storage and loading time with RDF/XML format.

(c) Testing and comparing the experiments with a Collective renaming.

2. Semantic level:

   - Adding parameters and creating a module for semantic order in the RDF statements.
   - Generating the properties to validate the Semantic RDF normalization.
   - Testing all the datasets with the semantic level.

3. IRI level:

   - Integrating ontology alignment for vocabularies utilized in the RDF description.
   - Generating properties to validate the Entity RDF normalization.
   - Testing all the datasets with the IRI level.

7.2.2 Extended Statement Recommendation Format

Using the extended statement format, it is possible to give more information for the developers of RDF descriptions and/or RDF data providers. This extension of the statement (triple) proposes to establish an extended logical format completely separated from the serialization. In this way, the developers can have all the information about languages and data types utilized in the RDF description before implementation. We consider as good practice to have the overall information for the development, having all the constraints that users might need in their models.

7.2.3 Guidelines for Generating Normalized RDF Descriptions

Another important direction is to propose guidelines to be use for different kind of users (e.g., professors, developers, students, etc.). These guidelines establish how to develop an RDF normalized description starting from the user's requirements and IRIs evaluation, and also evaluate the normalization level for their models. Our current work provides a valid starting point for further research on using normalized RDF descriptions in different applications.

These guidelines coined with the RDF normalization framework can be also used as a tool for evaluating RDF descriptions developed in an educational environment. These results provide an overall idea of the abstraction and correct utilization of the data.

7.2.4 RDF Normalization using Ontology Inference Mechanisms

Normalized RDF descriptions may also refers to different concepts (derived of ontologies) that may generate knowledge duplication in the statements of the description. Some derived statements of an RDF graph G may be deduced from one or more ontologies in several ways. With the normalization using ontology inference mechanisms, we may generate new statements of the RDF Graph G inferred of the original ones to reduce redundant knowledge.

7.2.5 Ontology Normalization

Another possible future direction is to devise a OWL Normalization approach based on our study. It is possible to provide normalized ontologies or RDF(S) files with the normalization levels because ontologies and vocabularies are developed following the RDF standard. We can attempt to normalized existing vocabularies or ontologies in our R2RN framework and analyze the results to discover new challenges to fulfill a normalized ontology or vocabulary. However, such a normalized ontology is not a trivial task, since the normalization process has to consider the inference engine through declarations (statements) in the original ontology.

7.2.6 Plug-and-Play Pre-Processing Component

On the long run, we aim to utilize our normalization approach as a pre-processing phase to prepare and clean RDF files to be effectively and efficiently utilized in semantic-aware applications, namely similarity-based approximate querying, approximate pattern matching, and similarity-based versioning within online RDF databases [AÖD14, Gea04, Gea11].

Bibliography

[AAD$^+$09] Dimitra Alexopoulou, Bill Andreopoulos, Heiko Dietze, Andreas Doms, Fabien Gandon, Jörg Hakenberg, Khaled Khelif, Michael Schroeder, and Thomas Wächter. Biomedical word sense disambiguation with ontologies and metadata: automation meets accuracy. *BMC Bioinformatics*, 10(1):1–15, 2009.

[ABMP08] Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. Rdfa in xhtml: Syntax and processing. *Recommendation, W3C*, 7, 2008.

[ACM10] Marcelo Arenas, Mariano Consens, and Alejandro Mallea. Revisiting blank nodes in rdf to avoid the semantic mismatch with sparql. In *RDF Next Steps Workshop*, pages 26–27, 2010.

[AMdLS06] Eneko Agirre, David Martínez, Oier López de Lacalle, and Aitor Soroa. Two graph-based algorithms for state-of-the-art wsd. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 585–593, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[AMMH07] Daniel J Abadi, Adam Marcus, Samuel R Madden, and Kate Hollenbach. Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd international conference on Very large data bases*, pages 411–422. VLDB Endowment, 2007.

[AÖD14] Güneş Aluç, M Tamer Özsu, and Khuzaima Daudjee. Workload matters: Why rdf databases need a new design. *Proceedings of the VLDB Endowment*, 7(10):837–840, 2014.

[AV08] Danny Ayers and Max Völkel. Cool uris for the semantic web. *Woking Draft. W3C*, 2008.

[BB01] Dave Beckett and A Barstow. N-triples-w3c rdf core wg. `http://www.w3.org/2001/sw/RDFCore/ntriples`, 2001. [Accessed 15-09-2015].

[BBL11] D Becket and Tim Berners-Lee. Turtle-terse rdf triple language. `http://www.w3.org/TeamSubmission/turtle/`, 2011. [Accessed 15-03-2016].

[BC06] Christian Bizer and Richard Cyganiak. D2r server-publishing relational databases on the semantic web. In *Poster at the 5th International Semantic Web Conference*, pages 294–309, 2006.

[Bea08] Francois Belleau and et al. Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. *J. Biomedical Informatics*, 41(5):706–716, 2008.

[Bec04] Dave Beckett. Rdf/xml syntax specification (revised). `http://www.w3.org/TR/rdf-syntax-grammar/`, 2004. [Accessed 15-09-2015].

[BG14] Dan Brickley and Ramanathan V Guha. Rdf schema 1.1. *W3C Recommendation*, 25:2004–2014, 2014.

[BHAR07] Chris Bizer, Tom Heath, Danny Ayers, and Yves Raimond. Interlinking open data on the web. In *Demonstrations track, 4th european semantic web conference, innsbruck, austria*, 2007.

[BHBL09] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227, 2009.

[BHIBL08] Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee. Linked data on the web (ldow2008). In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 1265–1266, New York, NY, USA, 2008. ACM.

[BL89] Tim Berners-Lee. Information management: A proposal http://www.w3.org. *History*, 198(9), 1989.

[BL05] Tim Berners-Lee. Notation 3 logic. `http://www.w3.org/DesignIssues/Notation3.html`, 2005. [Accessed 15-09-2015].

[BL06] Tim Berners-Lee. Linked data-design issues. `https://www.w3.org/DesignIssues/LinkedData.html`, 2006. [Accessed 15-09-2015].

[BL09] Tim Berners-Lee. Linked data-the story so far. *Int. Journal on Semantic Web and IS*, 5(3):1–22, 2009.

[BLCC$^+$06] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the 3rd international semantic web user interaction workshop*, volume 2006. Athens, Georgia, 2006.

[BLF00] Tim Berners-Lee and Mark Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor.* HarperInformation, 2000.

[BLFM98] Tim Berners-Lee, Roy Fielding, and Larry Masinter. Rfc 2396: Uniform resource identifiers (uri): Generic syntax. *Status: DRAFT STANDARD*, 1998.

[BLHL$^+$01] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[BM09] Sean Bechhofer and A Miles. Skos simple knowledge organization system reference. `http://www.w3.org/2006/07/SWD/SKOS/reference/20090315/skos.rdf`, 2009. [Accessed 15-03-2016].

[BM12] Dan Brickley and Libby Miller. Foaf vocabulary specification 0.98. *Namespace document*, 9, 2012.

[Boo03] David Booth. Four uses of a url: Name, concept, web location and document instance. *Retrieved January*, 28:2003, 2003.

[Boo08] David Booth. Why uri declarations? a comparison of architectural approaches. In *IRSW*, 2008.

[BPSM$^+$08] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and Franois Yergeau. Extensible markup language (xml) 1.0, 2008.

[BSB08] Paolo Bouquet, Heiko Stoermer, and Barbara Bazzanella. *An entity name system (ens) for the semantic web*. Springer, 2008.

[BSG07] Paolo Bouquet, Heiko Stoermer, and Daniel Giacomuzzi. Okkam: Enabling a web of entities. *I3*, 5:7, 2007.

[BSMG06] Paolo Bouquet, Heiko Stoermer, Michele Mancioppi, and Daniel Giacomuzzi. Okkam: Towards a solution to the"identity crisis"on the semantic web. In *SWAP*, volume 201, 2006.

[BSNM08] P. Bouquet, H. Stoermer, C. Niederee, and A. Maña. Entity name system: The back-bone of an open and scalable web of data. In *Semantic Computing, 2008 IEEE International Conference on*, pages 554–561, Aug 2008.

[CDD$^+$04] Jeremy J Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83. ACM, 2004.

[Con07] Dan Connolly. Gleaning resource descriptions from dialects of languages (grddl). `https://www.w3.org/TR/grddl/`, 2007. [Accessed 30-03-2016].

[DS04] Martin Dürst and Michel Suignard. Internationalized resource identifiers (iris). Technical report, 2004.

[Fea13] Javier D Fernández and et al. Binary rdf representation for publication and exchange (HDT). *J. Web Semantics*, 19:22–41, 2013.

[FGM+99] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1. `http://www.hjp.at/doc/rfc/rfc2616.html`, 1999. [Accessed 21-03-2016].

[Fie03] Roy T Fielding. httprange-14. *Resolved, W3C Tag*, 2003.

[FLBC16] Bernadette Farias-Loscio, Caroline Burle, and Newton Calegari. Data on the web best practices. `https://www.w3.org/TR/dwbp/`, 2016. [Accessed 20-06-2016].

[GCGP10] Raúl García-Castro and Asunción Gómez-Pérez. Interoperability results for semantic web technologies using OWL as the interchange language. *J. Semantic Web*, 8(4):278–291, 2010.

[Gea04] Claudio Gutierrez and et al. Foundations of semantic web databases. In *PODS 2004*, pages 95–106. ACM, 2004.

[Gea11] Claudio Gutierrez and et al. Foundations of semantic web databases. *J. Comput. Syst. Sci.*, 77(3):520–541, May 2011.

[GJM09] Hugh Glaser, Afraz Jaffri, and Ian Millard. Managing co-reference on the semantic web. 2009.

[GL+92] Ramanathan V Guha, Douglas B Lenat, et al. Language, representation and contexts. *JOURNAL OF INFORMATION PROCESSING-TOKYO-*, 15:340–340, 1992.

[GLMD07] Hugh Glaser, Tim Lewy, Ian Millard, and Ben Dowling. On coreference and the semantic web. 2007.

[GMB08] Aurona Gerber, Alta Merwe, and Andries Barnard. *The Semantic Web: Research and Applications: 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008 Proceedings*, chapter A Functional Semantic Web Architecture, pages 273–287. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[Guh08] Ramanathan V Guha. Meta content framework: A whitepaper. Technical report, Apple Technical Report, 2008.

[Hal11] Harry Halpin. Sense and reference on the web. *Minds and Machines*, 21(2):153–178, 2011.

[Hal13a] Harry Halpin. *Social Semantics: The Search for Meaning on the Web*, chapter Architecture of the World Wide Web, pages 9–50. Springer US, Boston, MA, 2013.

[Hal13b] Harry Halpin. *Social Semantics: The Search for Meaning on the Web*, chapter The Semantic Web, pages 51–83. Springer US, Boston, MA, 2013.

[Hay04] Patrick Hayes. Rdf semantics. w3c recommendation 10 february 2004. `https://www.w3.org/TR/2004/REC-rdf-mt-20040210/`, 2004. [Accessed 21-03-2015].

[HB11] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.

[HD05] Andreas Harth and Stefan Decker. Optimized index structures for querying rdf from the web. In *Web Congress, 2005. LA-WEB 2005. Third Latin American*, pages 10–pp. IEEE, 2005.

[HG04] Jonathan Hayes and Claudio Gutierrez. Bipartite graphs as intermediate model for RDF. In *Int. Sym. Semantic Web*, pages 47–61. Springer, 2004.

[HH08] Patrick J Hayes and Harry Halpin. In defense of ambiguity. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 4(2):1–18, 2008.

[HH10] Harry Halpin and Patrick J Hayes. When owl: sameas isn't the same: An analysis of identity links on the semantic web. In *LDOW*, 2010.

[HHM$^+$10] Harry Halpin, Patrick J Hayes, James P McCusker, Deborah L McGuinness, and Henry S Thompson. When owl: sameas isnŠt the same: An analysis of identity in linked data. In *The Semantic Web–ISWC 2010*, pages 305–320. Springer, 2010.

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation.* Addison Wesley, 2nd edition, 2001.

[HP09] Harry Halpin and Valentina Presutti. An ontology of resources for linked data. In *LDOW*. Citeseer, 2009.

[HPS14] Patrick J Hayes and Peter F Patel-Schneider. Rdf 1.1 semantics. 2014. [Accessed 21-03-2015].

[HPUZ10] Aidan Hogan, Axel Polleres, Jürgen Umbrich, and Antoine Zimmermann. Some entities are more equal than others: statistical methods to consolidate linked data. In *4th International Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic (NeFoRS2010)*, 2010.

[HSBW13] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[HSP13] Steve Harris, Andy Seaborne, and Eric PrudŠhommeaux. Sparql 1.1 query language. *W3C Recommendation*, 21, 2013.

[IV98] Nancy Ide and Jean Véronis. Introduction to the special issue on word sense disambiguation: The state of the art. *Comput. Linguist.*, 24(1):2–40, March 1998.

[Jea13] Guoqian Jiang and et al. Using semantic web technology to support ICD-11 textual definitions authoring. *J. Biomedical Semantics*, 4:11, 2013.

[JGM07] Afraz Jaffri, Hugh Glaser, and Ian Millard. Uri identity management for semantic web data integration and linkage. In *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, pages 1125–1134. Springer, 2007.

[JGM08a] Afraz Jaffri, Hugh Glaser, and Ian Millard. Managing uri synonymity to enable consistent reference on the semantic web. 2008.

[JGM08b] Afraz Jaffri, Hugh Glaser, and Ian Millard. Uri disambiguation in the context of linked data. 2008.

[JW04] Ian Jacobs and Norman Walsh. Architecture of the world wide web, volume one. w3c recommendation 15 december 2004. `https://www.w3.org/TR/webarch/`, 2004. [Accessed 21-03-2016].

[KC92] Robert Krovetz and W. Bruce Croft. Lexical ambiguity and information retrieval. *ACM Trans. Inf. Syst.*, 10(2):115–141, April 1992.

[KC04] Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. `http://www.w3.org/TR/rdf-concepts/`, 2004. [Accessed 21-03-2015].

[Kea07] Amine Kerzazi and et al. A model-based mediator system for biological data integration. *Journées Scientifiques en Bio-Informatique*, pages 70–77, 2007.

[Kea08] Amine Kerzazi and et al. A semantic mediation architecture for RDF data integration. *SWAP*, pp. 3, 2008.

[Knu98] D. Knuth. *The Art of Computer Programming. Volume 3: Sorting and Searching.* Addison-Wesley, Massachusetts, pp. 780, 1998.

[KVS07] John Krogstie, Csaba Veres, and Guttorm Sindre. Integrating semantic web technology, web services, and workflow modeling: Achieving system and business interoperability. *Int. J. of Enterprise Information Systems*, 3(1):22–41, 2007.

[LAH$^+$09] Atif Latif, Muhammad Tanvir Afzal, Patrick Hoefler, Anwar Us Saeed, and Klaus Tochtermann. Turning keywords into uris: simplified user interfaces for exploring linked data. In *Proceedings of the 2nd international conference on interaction sciences: information technology, culture and human*, pages 76–81. ACM, 2009.

[Les86] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA, 1986. ACM.

[LGMF04] Jure Leskovec, Marko Grobelnik, and Natasa Milic-Frayling. Learning substructures of document semantic graphs for document summarization. In *LinkKDD Workshop*, pages 133–138, 2004.

[Lon15] Dave Longley. RDF dataset normalization. `http://json-ld.org/spec/latest/rdf-dataset-normalization/`, 2015. [Accessed 21-08-2015].

[LS99] Ora Lassila and Ralph R Swick. Resource description framework (RDF) model and syntax specification. `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/`, 1999. [Accessed 21-03-2015].

[LSWC98] Ora Lassila, Ralph R. Swick, World Wide, and Web Consortium. Resource description framework (rdf) model and syntax specification, 1998.

[M$^+$07] David Martínez et al. Supervised corpus-based methods for wsd. In *Word Sense Disambiguation*, pages 167–216. Springer, 2007.

[MA12] Ibrahim F Moawad and Mohammadreza Aref. Semantic graph reduction approach for abstractive text summarization. In *Computer Engineering & Systems (ICCES), 2012 Seventh International Conference on*, pages 132–138. IEEE, 2012.

[MBLF05] Larry Masinter, Tim Berners-Lee, and Roy T Fielding. Uniform resource identifier (uri): Generic syntax. 2005.

[Mih06] Rada Mihalcea. *Word Sense Disambiguation: Algorithms and Applications*, chapter Knowledge-Based Methods for WSD, pages 107–131. Springer Netherlands, Dordrecht, 2006.

[Mil95] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[MJB12] Pablo N Mendes, Max Jakob, and Christian Bizer. Dbpedia: A multilingual cross-domain knowledge base. In *LREC*, pages 1813–1817. Citeseer, 2012.

[MLTB93] George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. A semantic concordance. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, pages 303–308, Stroudsburg, PA, USA, 1993. Association for Computational Linguistics.

[MMM$^+$04] Frank Manola, Eric Miller, Brian McBride, et al. RDF primer. `http://www.w3.org/TR/rdf-primer/`, 2004. [Accessed 21-08-2015].

[Nav09] Roberto Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69, February 2009.

[Nea12] Marc-Alexandre Nolin and et al. Building an hiv data mashup using Bio2RDF. *Briefings in bioinformatics*, 13(1):98–106, 2012.

[NV05] R. Navigli and Paola Velardi. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(7):1075–1086, July 2005.

[NWC03] Hwee Tou Ng, Bin Wang, and Yee Seng Chan. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 455–462, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[Pea09] Jyotishman Pathak and et al. Lexgrid: a framework for representing, storing, and querying biomedical terminologies from simple to sublime. *J. of the American Medical Informatics Assoc.*, 16(3):305–315, 2009.

[Ped06] Ted Pedersen. Unsupervised corpus-based methods for wsd. *Word sense disambiguation: algorithms and applications*, pages 133–166, 2006.

[PLDP07] Sameer S. Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. Semeval-2007 task 17: English lexical sample, srl and all words. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 87–92, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[PRM$^+$11] Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, CONLL Shared Task '11, pages 1–27, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[PSHH$^+$04] Peter F Patel-Schneider, Patrick Hayes, Ian Horrocks, et al. Owl web ontology language semantics and abstract syntax. *W3C recommendation*, 10, 2004. [Accessed 21-08-2015].

[PST$^+$15] Christoph Pinkel, Andreas Schwarte, Johannes Trame, Andriy Nikolov, Ana Sasa Bastinos, and Tobias Zeuch. Dataops: Seamless end-to-end anything-to-rdf data integration. In *The Semantic Web: ESWC 2015 Satellite Events*, pages 123–127. Springer, 2015.

[RŠD$^+$10] Delia Rusu, Tadej Štajner, Lorand Dali, Blaž Fortuna, and Dunja Mladenić. Demo: Enriching text with rdf/owl encoded senses. In *Proceedings of the 2010 International Conference on Posters & Demonstrations Track-Volume 658*, pages 133–136. CEUR-WS.org, 2010.

[RT12] Giuseppe Rizzo and Raphaël Troncy. Nerd: A framework for unifying named entity recognition and disambiguation extraction tools. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 73–76, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[SCV11] Leo Sauermann, Richard Cyganiak, and Max Völkel. Cool uris for the semantic web. 2011.

[SHJJ09] Henry Story, Bruno Harbulot, Ian Jacobi, and Mike Jones. Foaf+ ssl: Restful authentication for the social web. In *Proceedings of the First Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009)*, 2009.

[SL13] Manu Sporny and Dave Longley. RDF graph normalization. `http://json-ld.org/spec/ED/rdf-graph-normalization/20111016/`, 2013. [Accessed 21-08-2015].

[SLK$^+$14] Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, and Niklas Lindstrom. Json-ld 1.0 a json-based serialization for linked data. `http://www.w3.org/TR/json-ld/`, 2014. [Accessed 21-08-2015].

[SNA12] Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer. Keyword-driven resource disambiguation over rdf knowledge bases. In *Semantic Technology*, pages 159–174. Springer, 2012.

[SR14] Guus Schreiber and Yves Raimond. Rdf 1.1 primer. *W3C Working Group Note*, 2014.

[ŠRD$^+$09] Tadej Štajner, Delia Rusu, Lorand Dali, Blaž Fortuna, Dunja Mladenić, and Marko Grobelnik. Enrycher: service oriented text enrichment. *Proc. of SiKDD*, 2009.

[Srl] Deep Blue Srl. Resist: Resilience for survivability in ist.

[Sta09] W3C Standards. Help and faq. `https://www.w3.org/Help/#webinternet`, 2009. [Accessed 21-03-2016].

[STC14] Khouloud Salameh, Joe Tekli, and Richard Chbeir. SVG-to-RDF image semantization. In *Similarity Search and Applications*, pages 214–228. Springer, 2014.

[Sto08] Heiko Stoermer. Okkam: Enabling entity-centric information integration in the semantic web. 2008.

[Tea09] Cui Tao and et al. A RDF-base normalized model for biomedical lexical grid. In *the 8th Int. Semantic Web Conference*, Series A RDF-base Normalized Model for Biomedical Lexical Grid, pp. 2, 2009.

[Tek16] J. Tekli. An overview on xml semantic disambiguation from unstructured text to semi-structured data: Background, applications, and ongoing challenges. *IEEE Transactions on Knowledge and Data Engineering*, PP(99):1–1, 2016.

[THTC$^+$15] Regina Ticona-Herrera, Joe Tekli, Richard Chbeir, Sébastien Laborie, Irvin Dongo, and Renato Guzman. Toward rdf normalization. In Paul Johannesson, Mong Li Lee, Stephen W. Liddle, Andreas L. Opdahl, and Oscar Pastor Lopez, editors, *Conceptual Modeling*, volume 9381 of *Lecture Notes in Computer Science*, pages 261–275. Springer International Publishing, 2015.

[THTCL16] Regina Ticona-Herrera, Joe Tekli, Richard Chbeir, and Sébastien Laborie. Resolving logical redundancies and physical disparities in rdf descriptions (under review). *Journal of Web Semantics: Science, Services and Agents on the World Wide Web (JWS)*, 2016.

[TRC$^+$13] Joe Tekli, Antoine Abou Rjeily, Richard Chbeir, Gilbert Tekli, Pélagie Houngue, Kokou Yetongnon, and Minale Ashagrie Abebe. Semantic to intelligent web era: building blocks, applications, and current trends. In *Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems, MEDES 2013*, pages 159–168. ACM, 2013.

[UNR$^+$14] Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Michael Röder, Daniel Gerber, Sandro Athaide Coelho, Sören Auer, and Andreas Both. Agdistis-graph-based disambiguation of named entities using linked data. In *The Semantic Web–ISWC 2014*, pages 457–471. Springer, 2014.

[VAGS06] Mark Van Assem, Aldo Gangemi, and Guus Schreiber. Conversion of wordnet to a standard rdf/owl representation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LRECŠ06), Genoa, Italy*, pages 237–242, 2006.

[Vea09] Denny Vrandecic and et al. RDF syntax normalization using XML validation. *Proc. of the SemRUs*, pp. 11, 2009.

[VKM07] Raphael Volz, Joachim Kleb, and Wolfgang Mueller. Towards ontology-based disambiguation of geographical identifiers. In *I3*, 2007.

[Wal01] M. Mitchell Waldrop. *The Dream Machine: J.C.R. Licklider and the Revolution That Made Computing Personal.* Viking Penguin, 2001.

[Wea03] Kevin Wilkinson and et al. Efficient rdf storage and retrieval in jena2. In *SWDB*, volume 3, pages 131–150, 2003.

[WKB08] Cathrin Weiss, Panagiotis Karras, and Abraham Bernstein. Hexastore: sextuple indexing for semantic web data management. *Proc. VLDB Endow.*, 1(1):1008–1019, 2008.

[WW06] Kevin Wilkinson and Kevin Wilkinson. Jena property table implementation, 2006.

[Yar92] David 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*, COLING '92, pages 454–460, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[ZV11] Wen Zhu and Summeet Vij. Extending SOA infraestructure for semantic interoperability. `http://fr.slideshare.net/bfmj4fj/extending-soa-infrastructure-for-semantic-interoperability-13620422`, 2011. [Accessed 15-01-2015].

Appendix A

Appendix

Merge Sort is a sorting algorithm that takes advantage of the ease of merging already sorted lists into a new sorted list. Algorithm 3 provides the pseudo-code of the overall process. It starts by recursively decomposing the list (of extended statements) to be merged, into equal halves (lines 2 - 6). If the obtained sub-lists are of cardinality (length) 1, then they are already sorted, otherwise the algorithms continues with the decomposition process (lines 7 and 8). Consequently, it then merges each pair of resulting single extended statement lists, by comparing corresponding extended statements and swapping them if the first should come after the second (lines 9-12). The merging process is recursively executed until at last two lists are merged into the final sorted list. Algorithm Merge Sort is of O($N \times logN$) worst-case complexity where N is the size of the list (e.g., number of statements in an RDF description) being sorted, and thus scales well to very large lists [Knu98] (e.g., very large RDF description).

Algorithm 7 MergeSort

```
Input: $ST^+$ //List of Extended Statements of the RDF Description to be sorted
Output: $ST^{+'}$ //Sorter list of Extended Statements of the RDF Description
    Variables: Left, Right //Temporary list exploited in the sorting process
 1: if $card(ST^+) \leq 1$ then
 2:     return $ST^+$ //A list of length 1 is already sorted
 3: for i=0, i < $\frac{card(ST^+)}{2} - 1$, i++ do
 4:     add $ST^+$[i] to Left_$ST^+$ //Decomposing $ST^+$ into two halves
 5: for i=$\frac{card(ST^+)}{2}$, i < $card(ST^+)$, i++ do
 6:     add $ST^+$[i] to Right_$ST^+$ //of about equal sizes
 7: Left_$ST^+$ = $MergeSort$(Left_$ST^+$) //Recursive decomposition process
 8: Right_$ST^+$ = $MergeSort$(Right_$ST^+$) //Recursive decomposition process
 9: if Right_$ST^+$.last_Statement $\leq_\Psi$ Left_$ST^+$.first_Statement then
10:     $ST^+$ = $Fusion$(Left_$ST^+$,Right_$ST^+$)// Merges the lists by comparing their elements (statements)
11: else
12:     $ST^+$ = $Append$(Left_$ST^+$,Right_$ST^+$)// Simply appends both lists, since the last element (statement)of the first
        list is $\leq$ the first element (statement) of the second
13: return $ST^+$
```

Algorithm 8 Fusion //*used in the MergeSort algorithm*

```
Input: Left_$ST^+$,Right_$ST^+$ //Two lists of extended statements to be merged
Output: $ST^{+'}$ //Output merged and sorted list
 1: while $card$(Left_$ST^+$) $> 0$ and $card$(Right_$ST^+$) $> 0$ do
 2:     if Left_$ST^+$.first_Statement $\leq_\Psi$ Right_$ST^+$.first_Statement //Comparing statements of both lists then
 3:         Append(Left_$ST^+$.first_Statement,L)
 4:         Left_$ST^+$ = Left_$ST^+$ - {Left_$ST^+$.first_Statement}
 5:     else
 6:         Append(Right_$ST^+$.first_Statement,$ST^{+'}$)
 7:         Right_$ST^+$ = Right_$ST^+$ - {Right_$ST^+$.first_Statement}
 8: if $card$(Left_$ST^+$) $> 0$ then
 9:     Append(Left_$ST^+$,$ST^{+'}$)
10: else
11:     Append(Right_$ST^+$,$ST^{+'}$)
12: return $ST^+$
```

Note that algorithm *Merge Sort* is stable and *not in place*:

- *Stable*: Maintains the order of elements with equal values,
- *Not in place*: requires auxiliary structures for data to be temporarily stored (i.e., temporary lists Left_ST^+ and Right_ST^+, and output list $ST^{+'}$ exploited in the main algorithm).

Also, the pseudo-code of our statements comparison operator ($\leq_\Psi$), defined following our statement sorting expression Ψ (cf. Section 7.2.2) is provided below.

Algorithm 9 Statement Comparator $\leq_\Psi$

```
Input: $st_1^+, st_2^+$ //Extended statements to compare
Output: $st^+$ //Select extended statement
1: if $st_1^+.ts < st_2^+.ts$ //where $IRI > BN$ then
2:     return $st_1^+$
3: else
4:     return $st_2^+$
5: if $st_1^+.s$ is lexicographically lesser than $st_2^+.s$ then
6:     return $st_1^+$
7: else
8:     return $st_2^+$
9: if $st_1^+.to < st_2^+.to$ //where $IRI > BN > literal$ then
10:    return $st_1^+$
11: else
12:    return $st_2^+$
13: if $st_1^+.to$ is not blank node then
14:    if $st_1^+.o$ is lexicographically lesser than $st_2^+.o$ then
15:        return $st_1^+$
16:    else
17:        return $st_2^+$
18: if $st_1^+.p$ is lexicographically lesser than $st_2^+.p$ then
19:    return $st_1^+$
20: else
21:    return $st_2^+$
```

Appendix B

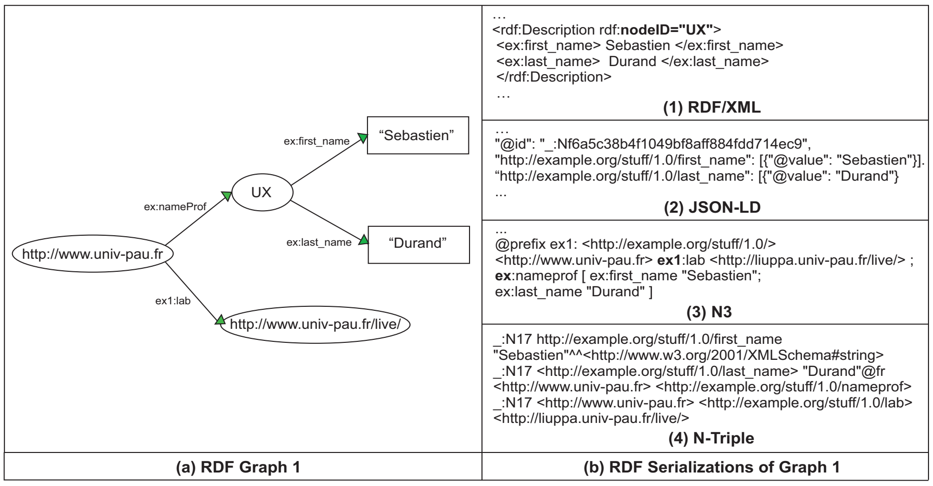
Résumé étendu

Introduction

Le Web Sémantique est un espace d'information qui consiste, d'une part, à lier les ressources du Web les unes aux autres, et, d'autre part, à leur donner du sens afin que les machines puissent les comprendre et les exploiter. Ce cadre permet à la recherche d'information d'être plus performante au même titre que les processus de gestion et d'échanges de données (e.g., recherche d'information intelligente, intégration de données, fusion, classification, etc.). Dans ce contexte, la technologie exploitée dans le domaine du Web Sémantique qui permet de connecter les ressources du Web entre elles est la suivante : RDF (Resource Description Framework), un standard du W3C (World Wide Web Consortium). De manière synthétique, une description RDF est formée d'un ensemble de triplet $< Sujet, Prdicat, Objet >$. Ces triplets forment un graphe RDF qui met en lumière les liens ou relations sémantiques entre différentes ressources.

Par exemple, le triplet suivant : $<$ `http://www.univ-pau.fr`, $ex1 : lab$, `http://liuppa.univ-pau.fr/live/` $>$ signifie que le sujet `http://www.univ-pau.fr`, identifié par son IRI (Internationalized Resource Identifier), dispose d'un laboratoire ($ex1$: lab) qui est référencé par une autre IRI `http://liuppa.univ-pau.fr/live/`. De nombreuses descriptions RDF de ce type qui contiennent elles-mêmes de multiples triplets sont actuellement disponibles en ligne grâce notamment aux projets de recherche qui traitent des Données Liées (Linked Data), tels que DBpedia, LinkedGeoData, Geonames, New York Times, etc. Ces initiatives autour des Données Liées Ouvertes (Linked Open Data - LOD) permettent aujourd'hui à tout à chacun (individus ou organisations) de partager des informations entre différentes communautés sur la base de triplets RDF.

Afin d'être stocké et exploité par une machine, ces triplets RDF sont sérialisés à l'aide de différents formats, tels que RDF/XML, N-Triple, Turtle, N3 ou bien JSON-LD. Comme



```
…
<rdf:Description rdf:nodeID="UX">
 <ex:first_name> Sebastien </ex:first_name>
 <ex:last_name>  Durand </ex:last_name>
 </rdf:Description>
 …
```

(1) RDF/XML

```
…
"@id": "_:Nf6a5c38b4f1049bf8aff884fdd714ec9",
"http://example.org/stuff/1.0/first_name": [{"@value": "Sebastien"}].
“http://example.org/stuff/1.0/last_name": [{"@value": "Durand"}
...
```

(2) JSON-LD

```
...
@prefix ex1: <http://example.org/stuff/1.0/>
<http://www.univ-pau.fr> ex1:lab <http://liuppa.univ-pau.fr/live/> ;
ex:nameprof [ ex:first_name "Sebastien";
ex:last_name "Durand" ]
```

(3) N3

```
_:N17 http://example.org/stuff/1.0/first_name
"Sebastien"^^<http://www.w3.org/2001/XMLSchema#string>
_:N17 <http://example.org/stuff/1.0/last_name> "Durand"@fr
<http://www.univ-pau.fr> <http://example.org/stuff/1.0/nameprof>
_:N17 <http://www.univ-pau.fr> <http://example.org/stuff/1.0/lab>
<http://liuppa.univ-pau.fr/live/>
```

(4) N-Triple

(b) RDF Serializations of Graph 1

Figure B.1: Exemple d'une description RDF.

illustré dans la figure B.1, une description RDF peut se représenter à l'aide d'un graphe (cf., Figure B.1.a) et peut être encodée à l'aide de différents langages (cf., Figures B.1.b.1, B.1.b.2, B.1.b.3 et B.1.b.4).

Dans certains scénarios (e.g., génération automatique, génération collaborative, intégration de données, etc.), les descriptions RDF peuvent être très verbeuses et peuvent également contenir de la redondance d'information. Ceci peut concerner à la fois la structure du graphe ou bien la sérialisation utilisée. Par exemple, considérons les deux graphes suivants : le graphe 1 dans la figure B.1.a et le graphe 2 dans la figure B.2 qui ont été créés par différents éditeurs. Ces deux graphes décrivent la même information : La ressource `http://www.univ-pau.fr` a un professeur qui s'appelle Sebastien Durand et qui dispose d'un laboratoire dont la référence est `http://www.univ-pau.fr/live/`. Néanmoins, leurs structures sont différentes comme il est possible de le constater visuellement dans ces deux figures. En effet, le graphe RDF de la figure B.2 contient des données dupliquées à la fois en ce qui concerne les nœuds mais aussi les arcs. Ceci a pour conséquence d'engendrer plus de triplets en comparaison avec le graphe RDF de la figure B.1.a. Bien entendu, la phase de sérialisation est impactée par ces redondances, sans compter les multiples variations d'écritures possibles au sein d'un même format pour décrire un triplet. Les éléments suivants permettent de se faire une idée des problèmes qui peuvent subvenir durant une sérialisation de descriptions RDF :

- La même ressource RDF peut être sérialisée de différentes façons (e.g., dans la figure B.1.b.1, nous avons utilisé l'attribut rdf:nodeID="UX" comme l'une des façons en RDF/XML de représenter un nœud de type "Blank Node". Ceci peut être effectué différemment dans d'autres langages de sérialisation.
- Le type de données ainsi que la langue d'une ressource RDF peut être spécifié ou non. Par exemple, dans la figure B.1.b.4, le type de données string est mentionné pour la ressource

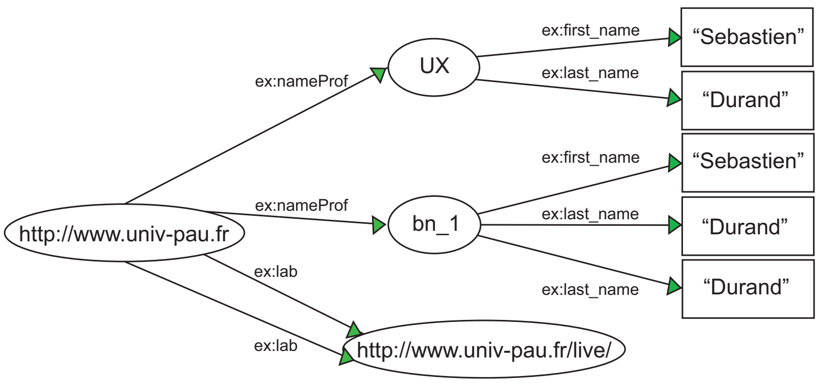


Figure B.2: Graphe RDF 2 décrivant la même information de la Fig. B.1.

Sebastien de la façon suivante "Sebastien"ˆˆ*xsd* : *string*. La langue fr est spécifiée quant à elle pour la ressource Durand comme ceci "Durand"@fr. Ces informations sur le type de données et la langue d'une ressource peuvent être omises dans d'autres formats de sérialisation.

- Différents espaces de nommage peuvent faire référence au même nom de domaine. Par exemple, le nom de domaine `http://xmlns.com/foaf/1.0/` dispose du nom d'espace de nommage "ex" dans la figure B.1.b.1 alors qu'il est nommé "ex1" dans la figure B.1.b.3.

Comme nous l'avons exposé précédemment, actuellement les individus ou les organisations s'échangent des données liées les unes aux autres, e.g., pour créer de nouvelles opportunités en terme de projets et/ou d'applications, pour favoriser entre autres le commerce en ligne avec des données supplémentaires pour les clients, pour accélérer les progrès scientifiques dans la gestion ou la recherche des données, etc. Dans ce contexte d'intégration et de fusion d'informations d'une ou de plusieurs ressources décrites par des communautés différentes, ces problèmes de redondances ou de variations d'encodage ne peuvent qu'être renforcés.

Par exemple, au sein d'une même description RDF, l'intégration de données au sujet de la ressource "Luxembourg" (voir la figure B.3) fourni par des communautés différentes, telles que DBpedia et Geonames, ne peut qu'accroître ce phénomène de redondances et de variations de sérialisation. De plus, ce bruit a un impact négatif sur le stockage ou bien le temps de chargement d'une description RDF. En effet, la description "RDF Graph X + RDF Graph Y + RDF Graph Z" prendra plus d'espace de stockage et de chargement qu'une description dépourvue d'informations redondantes ou inutiles.

Considérons maintenant le sous-graphe de la figure B.4. Nous pouvons constater des données redondantes tenant compte des ambiguïtés sémantiques ou encore des multiples IRI pointant vers la même information :

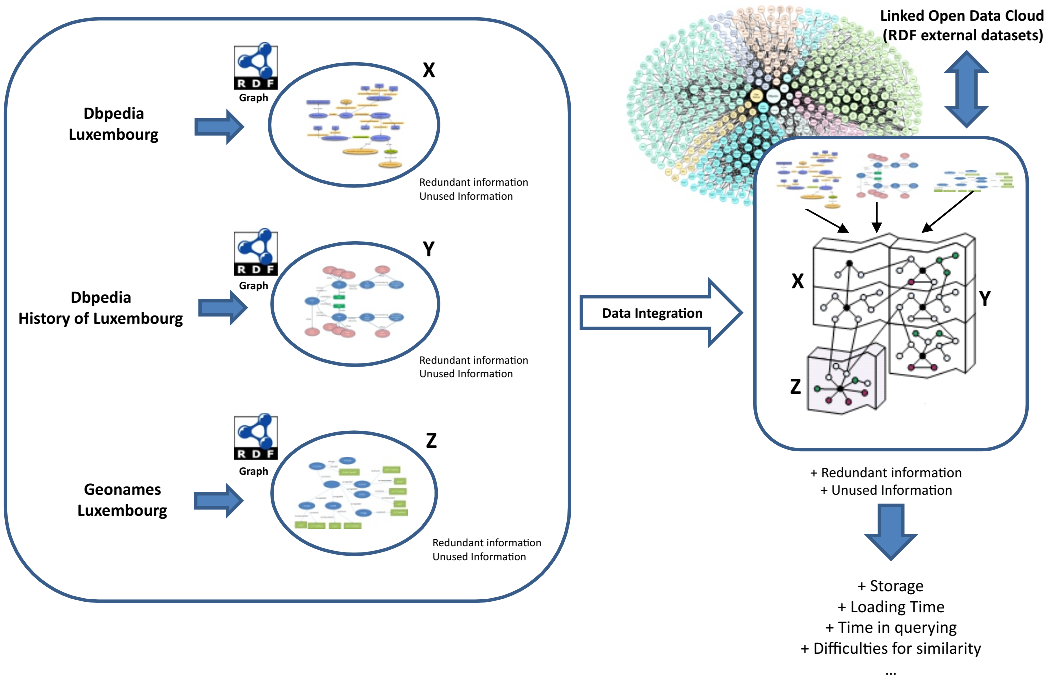


Figure B.3: Exemple de Données Liées sur la ressource "Luxembourg".

- Pour une ressource RDF, la déclaration du type ou de la langue peut faire référence à différentes valeurs primitives (e.g., chaîne de caractères, entier...) mais la signification sera équivalente. Par exemple, dans la figure B.4, la langue es est mentionnée dans l'expression "Luxembourgo"@es et la langue en est indiquée également "Luxembourg"@en. Nous pouvons donc constater que le nom de la ressource peut être dupliqué même si l'on évoque dans les deux cas la ville de Luxembourg.

- Différentes IRI peuvent faire référence à la même ressource RDF (e.g., dans la figure B.4, l'IRI `http://dbpedia.org/resource/Luxembourg` se base sur DBpedia mais l'IRI `http://sws.geonames.org/2960313` se réfère à Geonames. Ces deux IRI font référence à la même ville : Luxembourg.

Ces duplications ou variations d'encodage qui peuvent subvenir soit sur la structure du graphe RDF ou bien sur sa sérialisation doivent être prises en compte lorsqu'une description est à traiter. Ne pas la traiter aura forcément un impact négatif sur le développement ou le déploiement de bases de données RDF, ceci incluant le stockage, l'interrogation, le temps de chargement et de traitement, la mesure de similarité, l'appariement, l'alignement et le versionning pour ne citer que ces exemples.

Dans cette thèse, nous défendons une approche autour de la normalisation de descriptions RDF, c'est-à-dire épurer au maximum une description RDF. Dans la partie suivante, nous identifions nos principaux objectifs ainsi que nos contributions.

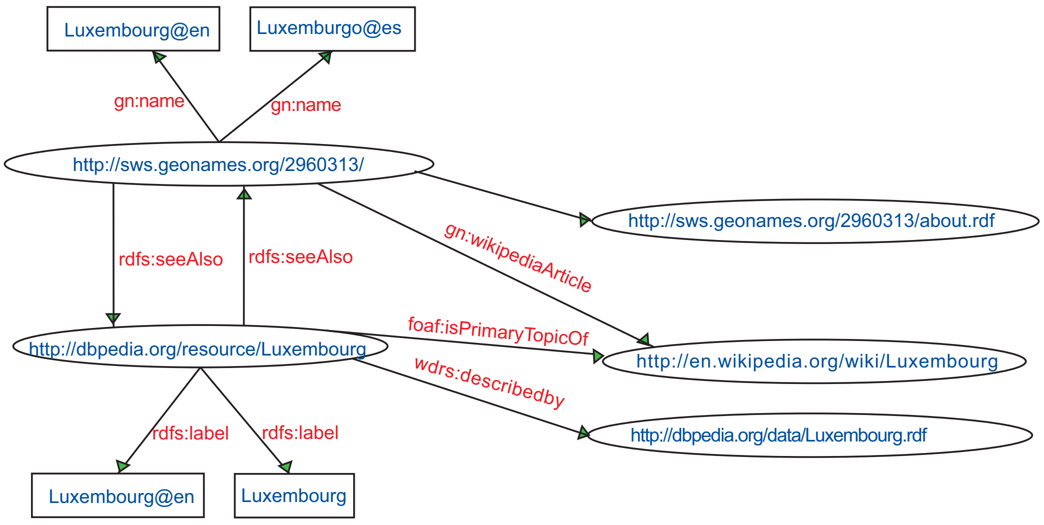

Luxembourg@en
Luxemburgo@es
gn:name
gn:name
http://sws.geonames.org/2960313/
http://sws.geonames.org/2960313/about.rdf
gn:wikipediaArticle
rdfs:seeAlso
rdfs:seeAlso
http://dbpedia.org/resource/Luxembourg
foaf:isPrimaryTopicOf
http://en.wikipedia.org/wiki/Luxembourg
wdrs:describedby
http://dbpedia.org/data/Luxembourg.rdf
rdfs:label
rdfs:label
Luxembourg@en
Luxembourg

Figure B.4: Sous-graphe d'intégrations de données dans la Fig. B.3

Nos objectifs de recherche

L'objectif de cette thèse consiste à résoudre les problèmes liés à la redondance d'informations ainsi que les variations d'encodage des descriptions RDF. Pour ce faire, nous avons défini un cadre de normalisation de descriptions RDF, nommé R2NR, qui permet de transformer n'importe quelle description RDF en une description RDF optimisée (normalisée). La sortie de ce processus peut être adaptée selon que le domaine d'application où sera appliqué la normalisation nécessite une optimisation en terme de stockage, comme la compression de la description par exemple.

Notre approche R2NR cible une normalisation de descriptions RDF selon les objectifs suivants :

- Eliminer les redondances au sein de la description (ceci est un objectif prioritaire puisqu'en optimisant la structure, on favorisera le traitement des requêtes, l'alignement et le versionning). Cet objectif se focalise sur la structure du graphe RDF.
- Eliminer les redondances ainsi que certaines structures d'encodage au sein du fichier RDF. Cet objectif se focalise sur l'écriture de la description RDF afin d'optimiser son stockage ainsi que son temps de chargement.
- Prouver que le processus de normalisation est : (i) valide selon un ensemble de propriétés, (ii) flexible et adaptable en fonction des critères d'un utilisateur ou des prérequis liés à une application d'un domaine métier en particulier, et (iii) efficace sur tout type de descriptions RDF contenant de quelques triplets RDF jusqu'à des milliers.

Nos contributions dans cette thèse

Selon les objectifs cités ci-dessus, les contributions de cette thèse sont les suivantes :

1. Normalisation syntaxique de descriptions RDF

   Ce défi qui consiste à normaliser des descriptions RDF a déjà été envisagé dans la littérature, néanmoins il n'a pas été atteint dans son ensemble. En effet, les solutions existantes restent partielles car elles ne considèrent pas tous les aspects syntaxiques liés aux descriptions RDF, comme par exemple la duplication de "Blank Nodes" ou encore les espaces de noms non-utilisés. Afin de réaliser une normalisation syntaxique de descriptions RDF, nous avons fourni :

   - Des définitions formelles des concepts exploités dans notre approche R2NR selon un modèle mathématique. Ces définitions systématiques et détaillées pour chaque élément de notre contribution incluent de surcroit des règles, des fonctions, des opérateurs avec les propriétés et les preuves associées.
   - Des algorithmes qui permettent une élimination de la redondance d'information au sein d'une description RDF, que ce soit au niveau du graphe ou bien au niveau de son encodage.
   - Des preuves permettant de valider qu'une description RDF normalisée préserve toute les informations initialement spécifiées par un éditeur.
   - Une approche personnalisable qui permet d'adapter la sortie de la normalisation de descriptions RDF en fonction de l'application cible (e.g., optimiser le stockage et/ou le temps de chargement).

2. Normalisation sémantique de descriptions RDF

   En plus de la normalisation syntaxique de descriptions RDF, nous avons étendu son champ d'actions afin de traiter les aspects liés à la sémantique des données. En effet, au-delà de la considération d'éléments dupliqués qui concernent une même ressource avec son identifiant, nous avons étendu notre approche R2NR pour que celle-ci traite d'éléments sémantiquement équivalents. Afin d'ajouter ce niveau sémantique à notre proposition R2NR, nous avons fourni :

   - Un ensemble de règles et de fonctions mathématiques qui permettent de résoudre les duplications d'informations en levant des ambiguïtés sémantiques identifiées entre certaines ressources.
   - Des algorithmes qui analysent le sens associé aux données issues d'une description RDF et élimine les redondances lorsque certains de ces éléments décrivent la même information.

3. Normalisation des IRI exploitées au sein d'une description RDF

   Nous avons proposé également de normaliser les IRI exploitées au sein d'une description RDF. Au sein du Web Sémantique, les IRI sont la base de descriptions d'une information. Il s'agit très souvent d'un identifiant qui sera exploité pour lui associer par la suite d'autres informations. Etant donné que différentes communautés peuvent décrire une même information (nous avons vu le cas précédemment avec la ville de Luxembourg), actuellement ces communautés attribuent chacune un identifiant différent pour décrire cette même donnée. Par conséquent, lors d'une fusion de plusieurs descriptions produites par différentes communautés, une description RDF va contenir différentes IRI décrivant une même information. Notre dernière contribution a donc consisté à normaliser ces différentes IRI décrivant un même concept. Pour ce faire, nous avons fourni :

   - Un ensemble de règles et de fonctions mathématiques qui permettent de résoudre les duplications d'IRI que ce soit au niveau de la structure du graphe RDF mais aussi au niveau de son encodage.
   - Des algorithmes qui identifient les similarités entre IRI afin d'éliminer leurs redondances au sein d'une description RDF.

4. Un prototype RDF2NormRDF

   Pour valider nos contributions, nous avons développé un outil nommé RDF2NormRDF. Il existe actuellement une version en ligne (`http://sigappfr.org/spider/research-projects/towards-rdf-normalization/`) et une autre disponible hors ligne. Cet outil a permis de tester et d'évaluer notre approche selon les critères suivants :

   - Efficacité : mesurer la qualité de détection et d'élimination des informations redondantes.
   - Performance : évaluer le temps d'exécution de notre approche de normalisation, le temps de chargement de la sortie produite ou bien sa taille mémoire.

   Nous avons présenté dans cette thèse une évaluation complète et détaillée de notre approche. Nous l'avons bien évidemment comparé à d'autres approches existantes. Les résultats ont été publiés dans les actes de la conférence ER 2015 (34th International Conference on Conceptual Modelling) et une version journal est en cours de soumission pour JWS (Journal on Web Semantics).

Structure du manuscrit

Je propose de décrire chaque chapitre dans ce qui suit afin de dresser un panorama du contenu de la thèse.

Chapitre 2

Dans ce chapitre, nous présentons le Web Sémantique ainsi que tous les concepts importants qui y sont rattachés et que nous utiliserons pour cette thèse. Il est primordial de bien comprendre les principes ainsi que les technologies existantes dans ce domaine pour bien appréhender les défis liés à la normalisation de descriptions RDF. En quelques mots, le Web Sémantique est lié à deux motivations principales :

- La première consiste à ce que quiconque peut décrire n'importe quelle information sur n'importe quel type de sujets. Ces descriptions doivent être distribuées, compréhensibles par les machines et liées les unes aux autres ceci afin d'enrichir une description.
- La seconde doit permettre de publier et de rechercher des descriptions par n'importe quel utilisateur.

La question principale des chercheurs a donc été de trouver ce moyen de publication et de partage de l'information, comme on peut le faire traditionnellement sur le Web des documents mais au sujet de connaissances. Dans ce chapitre, nous avons donc retracé l'histoire du Web (WWW) avec celle du Web Sémantique. Nous avons également décrit les concepts et technologies principales issues du Web Sémantique qui seront utilisées dans ce mémoire, telles que les IRI et RDF. Enfin, nous présenterons l'impact de ces technologies aujourd'hui sur le courant lié aux Données Liées (Linked Data).

Chapitre 3

Ce chapitre présente des scénarios qui exploitent des descriptions RDF au sujet par exemple de l'Université de Pau et du Luxembourg. Chaque scénario à pour objectif d'illustrer les problèmes de la non-utilisation ou bien de la duplication de données au sein d'une description RDF. Nous avons catégorisé ces problèmes en quatre niveaux :

- *Redondances logiques :* Plusieurs triplets RDF peuvent décrire la même information.
- *Redondances physiques :* Différents encodages avec différentes possibilités d'écriture pour chaque format peuvent décrire une même description RDF.
- *Ambiguïtés sémantiques :* Différents concepts décrivent sémantiquement la même information.
- *Divergences des IRI :* Différents identifiants de ressources peuvent décrire la même donnée.

Chaque niveau de problèmes est décrit en détail et illustré avec des exemples. Ces exemples serviront par la suite pour démontrer le processus de normalisation de descriptions RDF.

Chapitre 4

De plus en plus de descriptions RDF sont maintenant disponibles sur le Web. Les impacts liés au développement des technologies issues du Web Sémantique ou bien de courant comme celui des Données Liées ont conduit à ce qu'une multitude d'applications ont vu le jour. Ces applications exploitent bien évidemment des descriptions RDF pour afficher, partager et rechercher de l'information.

Ces descriptions peuvent contenir des redondances puisque différents utilisateurs les ont réalisés. Ces duplications de données sont à la base de la motivation de cette thèse qui consiste à nettoyer les descriptions RDF.

Ce chapitre présente donc la normalisation de descriptions RDF, sous un aspect pour le moment purement syntaxique. Nous commençons par introduire des définitions utiles pour poser le cadre de la normalisation. Nous définissons ensuite des fonctions, des règles ainsi que des opérateurs qui permettent de normaliser une description RDF. Nous discutons de l'état de l'art au sujet de la normalisation et des choix qui ont influencé la spécification de certains concepts utiles à la normalisation de descriptions RDF. Afin de valider plus tard ce cadre, nous spécifions un ensemble de propriétés qu'une description RDF normalisée doit satisfaire.

Chapitre 5

Comme nous le voyons dans le chapitre précédent, la normalisation de descriptions RDF a déjà été traitée pour différents types d'applications comme la représentation des connaissances, intégration de données, la théorie des graphes... Cependant, toutes ces approches se focalisent sur des problèmes qui concernent la syntaxe de la description RDF, laissant de côté les problèmes liés à l'ambiguïté sémantique ou bien ceux liés à la variabilité des IRI au sujet d'une même ressource.

Dans ce chapitre, nous présentons une extension de la normalisation de descriptions RDF en y intégrant des solutions permettant de lever ces problèmes d'ambiguïté et de variabilité des IRI tant au niveau logique (c'est-à-dire sur le graphe de description) qu'au niveau physique (c'est-à-dire au sein de l'encodage de la description). Pour ce faire, à l'instar du chapitre précédent, nous définissons des fonctions ainsi que des règles qui permettront une normalisation de descriptions RDF de plus haut niveau, c'est-à-dire tenant compte de la sémantique de l'information. Nous présentons les travaux de la littérature qui traitent de l'ambiguïté sémantique et de l'identité des IRI afin de mieux comprendre leur impact sur la duplication de données. Nous détaillons bien sur toute l'approche de normalisation et nous montrons le lien entre normalisation syntaxique et sémantique. Nous concluons ce chapitre par une comparaison de notre proposition avec les approches existantes.

Chapitre 6

Dans les deux chapitres précédents, nous avons présenté une approche de normalisation de descriptions RDF. Au sein de ce chapitre, nous présentons notre prototype, des mesures ainsi que l'environnement expérimentales que nous avons mis en place autour de ce prototype, et les résultats issus d'une évaluation. Nos expérimentations ont mis en lumière l'efficacité ainsi que la performance de notre prototype et approche, entre autres sur le temps de chargement de descriptions RDF dans le framework JENA ainsi que sur la taille mémoire des descriptions RDF normalisées. Nous avons comparé ces résultats avec d'autres méthodes, telles que JSON-LD ou HDT.

Conclusion

Dans cette thèse, nous avons proposé et évalué notre proposition de normalisation de descriptions RDF. Notre processus de normalisation élimine les redondances d'informations au sein d'une description que ce soit au niveau de la structure du graphe de description mais aussi au niveau de l'encodage de cette description. Nous avons défini formellement la normalisation de descriptions RDF à l'aide de règles, de fonctions, d'opérateurs, de propriétés et de preuves qui nous permettent de valider théoriquement notre contribution dans ce domaine. Bien évidemment, nous avons établi une analyse de l'état de l'art, notamment les méthodes de normalisation existantes tout en mettant en évidence leurs propriétés et leurs limites.

Dans ce qui suit, nous présentons quelques perspectives à notre travail tant au niveau de l'approche de normalisation elle-même que sur son potentiel impact sur d'autres technologies issues du Web Sémantique. En effet, ces perspectives concernent des pistes d'améliorations possibles de la normalisation de descriptions RDF, la normalisation d'ontologies, la normalisation à des fins d'apprentissage...

Dans les chapitres 4 et 5, nous avons montré que la normalisation de descriptions RDF peuvent s'opérer sur 3 niveaux : syntaxique, sémantique et sur les IRI. Nous avons également précisé qu'elle agit tant sur le graphe de description RDF que sur son encodage. La normalisation pour chacune de ces dimensions peut être améliorée :

- Niveau syntaxique :
  - Ajouter le traitement de la réification pour l'évaluation
  - Engendrer encore plus de sérialisations RDF au sein du prototype et évaluer les résultats en terme d'espace de stockage ou bien de chargement des descriptions RDF/XML normalisées.
  - Tester et comparer les expérimentations avec un renommage collaboratif.

- Niveau sémantique :
  - Ajouter des paramètres et créer un module pour ordonner chaque triplet RDF.
  - Générer des propriétés afin de valider la normalisation RDF au niveau sémantique.
  - Tester tous les jeux de données au niveau sémantique.
- IRI :
  - Intégrer l'alignement d'ontologie dans notre processus de normalisation.
  - Définir des propriétés pour valider la normalisation d'entités RDF.
  - Tester tous les jeux de données pour en démontrer son impact au niveau des IRI.

Une autre piste d'amélioration de notre travail consiste à proposer un guide qui sera exploité pédagogiquement par différents utilisateurs (e.g., enseignants, développeurs, étudiants, etc.). Ce guide doit pouvoir aider un utilisateur à produire une description RDF normalisée. Ce guide pourrait également être utilisé pour pouvoir évaluer ses propres descriptions RDF et notamment répondre aux questions suivantes : "Ma description RDF contient-elle de l'information redondante ?", "Ma description RDF peut-elle être encodée de façon plus optimisée ?"...

Dans cette thèse, nous avons normalisé des descriptions RDF. Ces descriptions font souvent référence à de multiples concepts (définis au sein d'ontologies). Le mécanisme de raisonnement sur ces ontologies peut en déduire de nouveaux triplets RDF qui, combiné à plusieurs ontologies, peut engendrer des duplications d'informations. Notre processus de normalisation peut tenir compte du mécanisme d'inférence sur les ontologies afin d'éliminer ces informations dupliquées.

Une ontologie est décrite à l'aide du formalisme RDF. Bien que cela nécessiterait d'intégrer de nouveaux vocabulaires et de prendre en considération les mécanismes d'inférence, il est tout à fait envisageable de normaliser des ontologies ou bien des descriptions RDF(S) à l'aide de notre travail dans cette thèse.

Pour finir, notre proposition de normalisation de descriptions RDF pourrait être une pré-phase incontournable pour préparer et nettoyer des descriptions RDF. Cette pré-phase serait un prérequis au calcul de similarité entre descriptions, au filtrage par motif ainsi qu'à la gestion de versions pour les bases de données RDF en ligne.