Model-driven Generation of Semantic Web Applications

Master’s Thesis, February 1, 2011

Michel van Tol
Model-driven Generation of Semantic Web Applications

THESIS

submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE
TRACK INFORMATION ARCHITECTURE

by

Michel van Tol
born in Rotterdam, The Netherlands

Web Information Systems Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
http://wis.ewi.tudelft.nl
Model-driven Generation of Semantic Web Applications

Author: Michel van Tol
Student id: 1263226
Email: michelvantol@gmail.com

Abstract

An increasing amount of datasets that can be combined with other datasets is emerging. Particularly the Semantic Web allows creating links between datasets, which increase the value of the dataset that is being linked. Finding links between datasets, as well as capturing the data inside another dataset that is of interest is a process for which tools have yet to be perfected. Often the process of enriching data involves understanding how to access the data, and finding clues how to use auxiliary data to add value to the original dataset.

This Thesis document proposes a framework for visually specifying the enrichment of RDF data. The framework is called RDF Gears, and allows incremental enrichment of RDF datasets. The platform provides functionality for composing enrichment flows from individual enrichment steps. The enrichment flow describes the order in which RDF data will be gathered from remote RDF data-sources. After specifying the flow, the enrichments can be applied to the RDF dataset. RDF Gears also provides functionality for creating and reusing enrichment steps and (nested) flows.

The approach discussed in this document shows the potential of specifying RDF enrichments incrementally. Also the possibility for abstracting Semantic Web technologies using the visual programming paradigm is shown. The main deliverable of this Master Thesis is the RDF Gears framework.

Thesis Committee:

Chair: Prof.dr.ir. Geert-Jan Houben, Faculty EEMCS, TU Delft
Committee Member: Dr. Laura Hollink, Faculty EEMCS, TU Delft
Committee Member: Dr. Andy Zaidman, Faculty EEMCS, TU Delft
Preface

I hereby present you the results of my Master’s Thesis Project, which I have been working on for about 10 months. After writing this thesis document, I realized the magnitude of the work that has been performed. In this preface I would like to thank the people who helped me to achieve this goal.

This project could not have been completed successfully without the guidance, support and feedback of a number of people I would like to thank now. First I would like to thank my supervisor, Geert-Jan Houben, for his inspiring and motivating attitude, as well as his critical view. Secondly, I would like to thank Jan Hidders, for bringing inspiration through long and fruitful discussions, as well as for providing me with valuable feedback on my Thesis document. Thirdly, I would like to thank Laura Holink for providing feedback during my Master’s Thesis project. Finally, I would like to thank Andy Zaidman for being the external committee member for the Thesis presentation.

I also would like to express many thanks to fellow graduate students and lab mates Jenna Willis and Eric Feliksik for providing a critical view and constructive feedback, as well as informal lunch meetings. Further, I would like to thank Boukje Sprenger for proofreading my thesis document.

Last but not least I would like to thank all of my family and friends for supporting me along this journey.

I hope you will enjoy reading my work!

Michel van Tol
Ridderkerk, the Netherlands
February 1, 2011
# Contents

Preface iii

Contents v

List of Figures vii

List of Tables ix

I Context 1

1 Introduction 3
   1.1 Background and motivation .......................... 3
   1.2 Problem definition and goal .......................... 4
   1.3 Document structure ................................. 5

2 Semantic Web and Linked Data 7
   2.1 Semantic Web ....................................... 7
   2.2 Linked Data ......................................... 11
   2.3 Conclusion .......................................... 13

3 Semantic enrichment 15
   3.1 General description of semantic enrichment ............ 15
   3.2 Related topics .................................... 16
   3.3 Presentation of the enrichment process ............... 18
   3.4 Requirements ..................................... 19
   3.5 Conclusion ........................................ 20

II RDF Gears 23

4 High-level design of RDF Gears 25
   4.1 Conceptual process of enriching RDF data in the RDF Gears framework 25
   4.2 Models driving the semantic enrichment process .......... 26
4.3 Executing the enrichment flow model . . . . . . . . . . . . . . . . . 33
4.4 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

5 Architecture of RDF Gears 37
5.1 Storage of RDF data . . . . . . . . . . . . . . . . . . . . . . . . . . 37
5.2 Core classes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
5.3 User interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
5.4 Enrichment engine . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
5.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43

6 Evaluation of RDF Gears 45
6.1 Comparing with other methodologies . . . . . . . . . . . . . . . . . 45
6.2 Extensibility of the RDF Gears framework . . . . . . . . . . . . . . . 46
6.3 Expressivity of the visual RDF enrichment language . . . . . . . . . 47
6.4 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49

III Conclusions and future work 51

7 Conclusions and Future Work 53
7.1 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
7.2 Summary per research question . . . . . . . . . . . . . . . . . . . . . 54
7.3 Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
7.4 Future work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56

Bibliography 59

Appendices 63

A Used RDF prefixes 65

B Third-party libraries 67
B.1 User interface libraries . . . . . . . . . . . . . . . . . . . . . . . . . 67
B.2 SPARQL and RDF libraries . . . . . . . . . . . . . . . . . . . . . . . 68

C Create enrichments with RDF Gears 69
C.1 Installing RDF Gears . . . . . . . . . . . . . . . . . . . . . . . . . . 69
C.2 Introducing the example dataset . . . . . . . . . . . . . . . . . . . . . 69
C.3 Enriching the example dataset . . . . . . . . . . . . . . . . . . . . . . 70
C.4 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
List of Figures

2.1 The Semantic Web stack ........................................ 8

3.1 Example workflow in Taverna ................................. 18

4.1 Overview of the 4 main phases of enriching RDF data ............ 25
4.2 Example of an input model ...................................... 27
4.3 Example of an output model .................................... 27
4.4 Example of a block representation in the UI .................... 28
4.5 Example of object typing applied to two instances within the Geonames RDF dataset .................................. 29
4.6 Example of an enrichment flow model .......................... 32
4.7 Example of DAG and a valid execution order ................... 33
4.8 Example of an enrichment flow containing more blocks ........... 33
4.9 Execution of an example enrichment flow ....................... 34

5.1 Overview of the architecture of RDF Gears ....................... 37
5.2 Overview of the RDF Gears GUI ............................... 40
5.3 Sequence diagram overview of enrichment engine ................. 41
5.4 Class diagram of currently available execution components ...... 42

6.1 Enrichment flow that is equivalent to the adapted Silk mapping .... 49

C.1 Example of a publication in dataset .............................. 70
C.2 Overview before importing RDF data ............................ 71
C.3 Overview before creating enrichment flow ......................... 71
C.4 Connected 'Geonames City lookup' block in enrichment flow .... 72
C.5 Overview of data enrichment and a summary of the results ......... 73
C.6 Summary of created vcard:geo triples .......................... 74
List of Tables

2.1 SPARQL query results ............................................. 11
4.1 Overview of port types and their representations in the UI ........ 28
4.2 Overview of the object types depicted in Figure 4.5 ............. 29
A.1 Overview of used RDF Prefixes ................................. 65
Part I

Context
Chapter 1

Introduction

The Semantic Web was invented by Tim Berners-Lee as a means to the end of constructing a machine-understandable web of data [6]. The foundation of the Semantic Web is to add meaning or semantics to information that can be found on the World Wide Web as well as to make computers support humans performing otherwise laborious tasks by searching or annotating this data. The Linked Data initiative is a closely related proposal initiated by Tim Berners-Lee, and is a description of how to publish data in a machine-readable way using the RDF data model [4]. The RDF data model allows creating datasets in any knowledge domain. Because of its properties, RDF is frequently used in the domain of medicine and biology. Also federal governments are interested in the possibilities offered by the Semantic Web.

At this moment in time, there is a need for a tool that allows RDF datasets to be made richer. This process should be supported by a framework that allows the user to understand how data is enriched, while at the same time not having the user bother about the details of the enrichment process.

This chapter formulates the main research goal of this Master’s Thesis Project. The main research problem is further decomposed in a number of research questions. Finally, the outline of this document is presented.

1.1 Background and motivation

The Semantic Web and in particular the Linked Data project are two initiatives that aim towards creating a more homogeneous information space, in which one could for example link one RDF dataset to another RDF dataset (or perhaps multiple RDF datasets).

Many online institutions currently make use of their own standards and APIs, which makes tasks such as gathering data depend on the approaches taken by the institution. Instead of promoting data silos that are created by a non-standard way of publishing data, fragmenting the web into separate compartments, a single set of rules and guidelines to publish data is proposed [8].

Linked Data principles propose a data model (the RDF data model) as well as principles that should be adhered to while publishing data [4]. These guidelines are used to create a cloud of data. Building on this infrastructure, an RDF dataset can be linked with other (usually more commonly used) RDF datasets.
1.2 Problem definition and goal

Since the Linked Data principles allow data to be accessed in a similar way, new possibilities for (re)using data arise. One example is the possibility to link RDF datasets to commonly used Linked Data datasets. The datasets within the Linked Data cloud describe a great variety of resources ranging from people, geographical data, organizations, but also more specific resources like scientific publications, medical data, data from the music and movie industry.

RDF datasets can be enriched with pointers to resources within the Linked Data cloud, to allow the RDF data to represent a more concisely specified and more globally understood set of resources. Apart from the aspect of creating new links, also literal data can be used to enrich RDF datasets.

We identified the need for enriching an RDF dataset, now we will describe the state-of-the-art in enriching RDF datasets. Currently, creating enriched RDF datasets needs the developer to thoroughly understand the principles of Semantic Web technologies, and perform the process of attaching additional data to the RDF dataset manually. Bizer et al. state that schema mapping and data fusion are challenges that have yet to be overcome [9]. We feel that providing a tool for better semantic enrichment can (partially) tackle this open issue. This thesis project aims at creating a platform in which enrichments can be executed without the need to understand the specifics of the auxiliary RDF datasets used for enriching, as well as being able to provide custom enrichments to this platform. The enrichment process should be represented in such a way that the user of the system understands how enrichments are performed.

1.2 Problem definition and goal

This Master’s Thesis document proposes a model-driven approach to enrich RDF datasets with RDF data from external datasets. The value of an RDF dataset can be enhanced by adding relevant data to it. The basic idea of dataset enrichment is to allow better querying (e.g. with SPARQL\textsuperscript{1} or SeRQL\textsuperscript{2}) as well as better browsing (e.g. with Semantic Web browsers such as Tabulator\textsuperscript{3}, Marbles\textsuperscript{4}, Disco\textsuperscript{5} or Explorator\textsuperscript{6}). By creating additional links, the RDF dataset will blend in better with the Linked Data cloud.

Enriching a dataset can often become a laborious task, as the user needs to understand the structure of the external data in order to make links towards it. The main idea of RDF Gears, the tool that is designed and implemented during the course of the thesis project, is to make RDF enrichments possible in a way the user understands what data is generated, abstracting SPARQL querying and other Semantic Web technologies as much as possible.

\textsuperscript{1}http://www.w3.org/TR/rdf-sparql-query/
\textsuperscript{2}http://www.openrdf.org/doc/sesame2/users/ch09.html
\textsuperscript{3}http://www.w3.org/2005/ajar/tab
\textsuperscript{4}http://marbles.sourceforge.net/
\textsuperscript{5}http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/
\textsuperscript{6}http://www.tecweb.inf.puc-rio.br/explorator
1.2.1 Research goal

The research reported in this Thesis document was coordinated at the TU Delft, as part of ongoing research in the Web Information Systems (WIS) group. The following main research goal was formulated:

**Research goal:** Investigate and construct a model-driven environment that allows enriching RDF data in an intuitive way, by adding additional information from external sources and without posing any substantial restrictions on the source RDF dataset.

Each of the chapters in this document aims at answering a particular research question. We constructed each of the following research questions as a means to determine whether we achieved the research goal. Each of the following chapters answers a specific question:

- Chapter 2: What is the Semantic Web and which standards are used to create Semantic Web applications? What is Linked Data and what are the benefits of using Linked Data?
- Chapter 3: What is semantic enrichment, and how can the process of semantic enrichment be facilitated?
- Chapter 4: What is the basic conceptual process of the RDF Gears framework to guide semantic enrichment? Which are the models that drive the process of creating a semantically enriched RDF dataset?
- Chapter 5: What is the architecture used by the RDF Gears framework to enrich RDF data?
- Chapter 6: How does the RDF Gears framework relate to other systems, technologies and approaches?

Each chapter concludes with a summary and an answer to the research question.

1.3 Document structure

This document is divided in three parts. Part I identifies the background knowledge that is used throughout this document. Chapter 1 introduces the research topic and the research goal. Chapter 2 contains a description of the Semantic Web and Linked Data. It shows the technologies behind the Semantic Web, as well as the practical use of Linked Data. Chapter 3 discusses semantic enrichment and related topics. A number of requirements are also presented in Chapter 3.

Part II elaborates on the RDF Gears platform. Chapter 4 shows the high-level design of the RDF Gears framework. The models and conceptual process that should be supported by the framework are explained in detail. The visual enrichment specification language is also introduced. In Chapter 5, the architecture of the RDF Gears
platform is outlined. The architecture of the complete platform is outlined first, then
the basic components are described in detail. Chapter 6 compares the performance of
RDF Gears to a number of related technologies and approaches. Also, the expressivity
of the visual enrichment specification language is evaluated.

Part III concludes this document. Chapter 7 reflects on the research goal and re-
search questions by discussing the main conclusions and contributions of this project.
Also future work will be discussed in this chapter.
Chapter 2

Semantic Web and Linked Data

This chapter introduces both the Semantic Web as well as the Linked Data initiative. Let us first describe the history of the Internet. The Internet has been evolving rapidly from the moment it began at the US Defense Advanced Research Projects Agency (DARPA) and became known as the ARPANET, in the 1960’s [21]. The World Wide Web has been invented in 1989 and 1990 by Tim Berners-Lee and Robert Cailliau at CERN [11]. The idea behind the original World Wide Web is called hypertext, and is based on documents pointing to other documents using so-called hyperlinks [5]. This structure allowed humans to find information and navigate to documents which are related to it. Since we understand the text of hyperlinks, we can understand how the linked document is related to the current document. However, the vast amount of documents on the Internet has complicated the task of finding the information we are looking for.

Machines could help us, but meaning (i.e. semantics) needs to be added to the data, so that the machines can decide which information might be helpful for humans [6]. The Semantic Web, as the initiative of creating a machine-understandable web of data is often called, mainly consists of resources, which may be documents, persons or other entities, and the relations between these resources. This chapter will first describe the Semantic Web in more detail, after which the concept of browsing is outlined. Finally, browsing through the Semantic Web is introduced.

2.1 Semantic Web

The Semantic Web can be called an answer to the ongoing expansion of the World Wide Web. The World Wide Web Consortium (W3C), of which Tim Berners-Lee is the founder, has been active in developing Semantic Web technology over the past decade. Tim Berners-Lee saw that the World Wide Web was mainly used for the storage of documents, therefore it was often called the Web of Documents. The Semantic Web aims at creating a Web of Data, which can be achieved by adding meaning or semantics to the Web of Documents. His idea is to create additional data and documents which are specifically aimed at machines [6]. The machines would use this data to collect information about the current context, since in the form of flat text computers are unable to determine what the text is about, unlike human users. Berners-Lee et al. state that "The dream behind the Web is of a common information space in which we
communicate by sharing information”[3]. The W3C has developed standards, which can all be retrieved at the W3C website\(^1\). Next will be a brief introduction of the Semantic Web stack and a selection of key technologies incorporated in the Semantic Web.

### 2.1.1 Semantic Web stack

The Semantic Web stack is a series of technologies which provide the foundation of the Semantic Web. The complete Semantic Web stack is shown in Figure 2.1.

![The Semantic Web stack](http://www.w3.org/2001/sw/)

**Figure 2.1: The Semantic Web stack**

(source: [http://www.w3.org/2001/sw/](http://www.w3.org/2001/sw/))

On top of the Semantic Web lies an interface, which enables the use of the Semantic Web technologies shown in Figure 2.1.

- **Trust** Trust is an important open issue that will need to be solved in order to provide data about the trustworthiness of information scattered on the web.

- **Proof** The proof component within the Semantic Web is still under research. Its main objective is to be able to reason over (inferred) RDF data and determine whether parts of it can be proven correct.

- **Unifying Logic** Unifying Logic has not been specified completely (yet), but the main purpose of the layer would be to create a mathematically sound data model that combines the data from different partial models (RDF, RDFS, SPARQL, RIF and OWL).

- **Crypto** Cryptography is the part of the Semantic Web stack used to encrypt and decrypt data that is being exchanged.

- **OWL** Ontologies allow the creation of a system of constraints based on the types of resources which are described by the data, and the properties that these resources have.
SPARQL  SPARQL is a query language used to combine RDF-based data from potentially different sources based on user-specified requirements.

RIF  The Rule Interchange Format is a format in which specific rules (mostly inference rules) can be exchanged. These rules can be applicable to RDFS schema or OWL ontologies.

RDFS  RDF Schema allows the creation of a system of types as well as hierarchies. It has a simple inference mechanism which can be used to validate subtypes of a type.

RDF  The Resource Description Framework is a framework for expressing relations between resources and data. RDF allows describing web-based resources by creating a directed and labeled graph model connecting resources (URIs) with other resources as well as literal data. RDF has a number of different representation formats, one of which is RDF/XML.

XML  XML is a data communication format commonly used because of its ability to store structured data. XML can be used as a carrier for RDF, but does not contain semantics itself.

URI/IRI  The Uniform Resource Identifier (URI) is used to identify and locate resources, which can be regular hypertext pages on the World Wide Web. As mentioned previously, RDF uses HTTP URIs to describe data about a particular entity.

2.1.2 Standards used in the Semantic Web

This section will elaborate on RDF and SPARQL, which have been discussed while elaborating on the Semantic Web stack.

RDF  RDF has been developed in 1999 and is meant to store data describing other data, which is called metadata [26]. RDF is a framework for specifying resources (things) and their relation to other resources. An RDF graph is a collection of triples, each of which consisting of three components, a subject, a predicate and an object. An RDF triple thus connects a subject with an object via a predicate [19]. Both subject and predicates are usually a URI, objects may be either a resource (a URI) or a literal value. An example of such a triple, where URIs of persons are intentionally omitted, could be:

John (subject) knows (predicate) Jane (object)

Similarly, John could also have other friends in this model, as well as contact information such as an e-mail address for instance. RDF hence is a format in which labeled, directed graphs are described. The previously described situation is shown in an XML serialization of RDF in Listing 2.1.
RDF allows modeling domain-specific knowledge about entities, by making use of vocabularies. Examples vocabularies are Friend-Of-A-Friend\(^2\) (FOAF), Semantically-Interlinked Online Communities\(^3\) (SIOC) and the Dublin Core\(^4\). FOAF allows describing personal relations and contact information [13], while SIOC allows describing the structure of an online community [7]. The Dublin Core is often used to describe properties of published media (e.g. a web page, a video, a book or a painting), such as its creator, title and nature, [16]. In Listing 2.1, the predicate foaf:knows is used. The relationship knows is thus part of the FOAF (Friend-Of-A-Friend) vocabulary. In RDF, resources are linked by predicates such as foaf:knows.

RDF datasets can make use of a set of custom predicates, which are specified explicitly within the same RDF dataset. In most cases however, common vocabularies are able to express the relations between resources in an RDF dataset. Using common vocabularies like FOAF, SIOC and Dublin Core eliminates the need for creating custom predicates. Vocabularies are often specified in the RDF Schema language (RDFS\(^5\)), which allows defining types and subtypes, as well as hierarchies within RDF data.

Resources within any RDF dataset can be connected to resources that are not specified within the same RDF dataset. This characteristic of the RDF framework enables connecting RDF datasets with each other. Linking RDF datasets is discussed in more detail in Section 2.2.

---

\(^2\)[http://www.foaf-project.org/]

\(^3\)[http://sioc-project.org/]

\(^4\)[http://dublincore.org/documents/dcmi-terms/]

\(^5\)Officially known as: RDF Vocabulary Description Language, [http://www.w3.org/TR/rdf-schema/]
SPARQL

The SPARQL technology has been defined as a mechanism to extract (useful) information from RDF data [14]. SPARQL consists of both a protocol and a query language. The protocol addresses the fact that RDF data should be accessible remotely, and provides an interface to this end, which is called a ‘SPARQL endpoint’. Syntactically, SQL and SPARQL look alike, but their semantics are different because of the data they operate on (relational databases versus RDF data).

A SPARQL endpoint is used for executing a query in the SPARQL query language. Such a SPARQL query is normally used to extract a subset of the RDF data that has been made accessible by the SPARQL endpoint [25]. RDF data can be stored in segments on a SPARQL endpoint using RDF graphs. An example of a SPARQL query is shown in Listing 2.2.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?fullName ?email
WHERE
{ ?x foaf:name ?fullName .
  ?x foaf:mbox ?email }
```

Listing 2.2: Example SPARQL query

The example SPARQL query requests all URIs where both foaf:name and foaf:mbox are defined. The FROM-clause is used to specify the RDF graph to retrieve RDF data from while executing the SPARQL query. In this case, we assume the RDF data about John Doe and Jane Doe, shown in Listing 2.1, is stored in the default RDF graph of the SPARQL endpoint, and thus can omit the FROM-clause from the SPARQL query. The SPARQL endpoint processes the SPARQL query (shown in Listing 2.2) by retrieving RDF triples for which both predicates are defined. The results of the query are shown in Table 2.1.

<table>
<thead>
<tr>
<th>fullname</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;John Doe&quot;</td>
<td>&quot;<a href="mailto:john.doe@example.com">john.doe@example.com</a>&quot;</td>
</tr>
<tr>
<td>&quot;Jane Doe&quot;</td>
<td>&quot;<a href="mailto:jane.doe@example.com">jane.doe@example.com</a>&quot;</td>
</tr>
</tbody>
</table>

Table 2.1: SPARQL query results

2.2 Linked Data

A sub-project within the Semantic Web is called the Linking Open Data project. Its aim is to create a vast interconnected graph of relations between resources within different domains or different datasets within the same knowledge domain. Linked Data is a way of publishing RDF datasets making these datasets publicly accessible and open for external linking.

The Linked Data Cloud is a composition of all RDF datasets that can be publicly accessed and used for the purpose of linking other datasets against. The process of

---

6Current status: [http://richard.cyganiak.de/2007/10/1od/lod-datasets_2010-09-22_colored.png](http://richard.cyganiak.de/2007/10/1od/lod-datasets_2010-09-22_colored.png)
linking datasets to the Linked Data cloud will be discussed in Chapter 3.

One domain where Linked Data has already proven its strengths is the area of life sciences. The Linked Open Drug Data\(^7\) initiative shows the interconnectedness of RDF life sciences datasets.

### 2.2.1 Linked Data principles

Tim Berners-Lee has stated a total of four rules which are vital for the Linked Data project \([4]\). They are described as follows:

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);
4. Include links to other URIs, so that more things can be discovered by others.

Including links to other URIs (rule 4) is particularly useful, because it states that resources within one RDF dataset should be linked to resources within another RDF dataset. By adding connections between resources in different datasets, one is essentially linking two RDF datasets.

### 2.2.2 Benefits of linking data

Linking data offers opportunities to the RDF data that is being linked. One of the main aims is to link the resources in an RDF dataset to resources in a remote RDF dataset (for example, linking a resource representing a news article about the city of Delft to the resource representing the city of Delft in the Geonames\(^8\) dataset).

Extending RDF data will allow the increase of search and navigation performance and enhanced SPARQL query capabilities. For instance, linking to resources in the DBpedia\(^9\) dataset creates the possibility of navigating from the original RDF dataset, through a resource in the DBpedia dataset, to a resource in one of the other Linked Data datasets \([2]\).

The added value from linking RDF data with a dataset in the Linked Data Cloud, is that it makes the linked RDF dataset more flexible in use. Three main uses of Linked Data that are exemplified by the DBpedia dataset according to Bizer \textit{et al.} are \([10]\):

1. Enhanced browsing and crawling of the Web of Data;
2. New possibilities for fusing datasets and creating mash-ups;
3. A system for annotating content on the Web of Documents.

Linked Data provides a set of guidelines and rules (such as the Linked Data principles, see Section 2.2.1). The platform that Linked Data offers makes clear that the separation of the Web into separate data silos represents a lack of a unifying set of principles for the identification, searching and the representation of data \([8]\).

\(^7\)http://esw.w3.org/HCLSIG/LODD

\(^8\)http://www.geonames.org/

\(^9\)http://www.dbpedia.org
2.3 Conclusion

The vision behind the Semantic Web was allowing machines to understand more deeply the information provided on the World Wide Web. In this chapter, we discussed RDF and SPARQL in detail. These two technologies will be referred to later in this document.

The Linked Open Data initiative was started to create a platform of linkable RDF datasets. Different datasets like DBpedia, Geonames, but also many domain-dependent datasets (life sciences, publications, entertainment and government data) are available to link other RDF datasets against. Linking RDF data to one or more datasets in the Linked Open Data cloud makes the linked RDF data more navigable and browsable, which increases the value of the RDF dataset.
Chapter 3

Semantic enrichment

Chapter 2 discussed the Semantic Web as well as Linked Data. In this chapter, a general description of semantic enrichment is given. From this description the functionality of the RDF Gears application should become clear. In this chapter we will also discuss related topics such as semantic interlinking and the semantic interlinking tool Silk, as well as inference rules. An overview of how the semantic enrichment process could be visualized is shown next. This chapter will end by stating a number of requirements that need to be fulfilled by a tool that provides semantic enrichment capabilities.

3.1 General description of semantic enrichment

We will use the term semantic enrichment in this document as a process where we start using a core RDF dataset (i.e. input RDF dataset), retrieve relevant data from external RDF datasets, and store the core RDF dataset and the retrieved RDF data in the enriched application RDF dataset (i.e. the output RDF dataset).

Semantic enrichment is the process of adding new RDF data to an initial RDF dataset. The initial RDF dataset is representable in the form of an RDF graph. The RDF graph is composed of a set of RDF triples. The data that is being added to the initial RDF dataset is also representable in an RDF graph, whereas the result of the semantic enrichment process is also an RDF graph that merges the initial RDF dataset with the RDF data retrieved during the enrichment process.

The basic idea of semantic enrichment as is discussed during the course of this document, is that we start with resources described by $x$ properties and manipulate this data in such a way that resources will have $x+1$ properties, or even more. In other words, we use the current knowledge about a specific resource, and infer more knowledge by finding clues in other RDF datasets (i.e. auxiliary datasets). While enriching RDF data, we are able to specify the name of the new property, so we have full control of how the new property will become part of our enriched application data.

Semantic enrichment is not merely a process of creating links between entities, but can also involve manipulating literal data such as converting units or currencies and other arithmetic operations. The format in which data was provided by another RDF dataset is not always the format that is the most universal way of expressing data. Semantic enrichment in my vision also deals with this issue.
Terms that come close to the description of semantic enrichment that was just given are ‘semantic data integration’, ‘semantic interlinking’ and ‘rule-based inference’. We will discuss each of these topics now.

3.2 Related topics

In this section we will elaborate on a number of topics closely related to the description of semantic enrichment that was just given in Section 3.1.

3.2.1 Semantic data integration

Semantic data integration is quite similar to the description of semantic enrichment that was just given. Semantic data integration is a process in which entities in a ‘global schema’ or ontology are mapped to entities in local data sources [12]. A number of relations have been discovered by Cruz and Rajendran [12], namely:

- One-to-one mapping: the global entity is equivalent to exactly one entity in the local datasources;
- One-to-null mapping: the global entity does not have an equivalent entity in the local datasources;
- Parent-child mapping: the parent entity is part of the global schema, while its children are part of the local datasources;
- One-to-many mapping: one global entity is mapped to a collection of entities in the local datasources;
- Many-to-one mapping: multiple global entities are mapped to a single entity in the local datasources.

Looking at the relations as described above, one can see that semantic data integration is a methodology for mapping entities to other entities, aiming for finding semantically similar entities. An interesting idea behind semantic data integration is that after the correct mappings between the global schema and local datasources are in place, user queries expressed on the global schema will be rewritten as a set of subqueries in terms of entities in the local datasources [12].

3.2.2 Semantic interlinking

The topic of semantic interlinking is a subset of semantic enrichment. Semantic interlinking aims at finding equivalent resources in different datasets, and link them together based on their properties. Finding equivalent resources in different datasets is only a part of the semantic enrichment problem, but having found the correct equivalences, finding extra information about the identified resource is a trivial task.

Semantic interlinking is a process that needs human intervention, because it is the human that needs to decide whether two resources are equivalent. An interesting problem in this domain is identified by Halpin and Hayes who state that even when two resources are linked via the \texttt{owl:sameAs} predicate (which states these two resources
are semiantically equivalent), they might not be equivalent at all (e.g. because some properties are defined for one resource, but not for the other) [15].

**Silk**  The Silk interlinker\(^1\) is a tool that creates semantic interlinkings. Silk is a framework for finding semantically equivalent resources in different datasets, and creating links for each pair of resources that is semantically equivalent [31].

The main feature of Silk is the Silk LSL (or Link Specification Language). This language allows the specification of a set of heuristics, which are used for deciding whether the conditions for semantic equivalence are satisfied (i.e. whether or not two compared resources are considered to be equivalent). Several properties of resources on both datasets can be compared using string similarity metrics, as well as recently implemented metrics for comparing two numbers (such as the population of a city), two dates and the geographical distance between two points\(^2\). Silk LSL allows the user to specify a number of these comparison metrics, and the conditions that need to be satisfied in order for the Silk interlinker to conclude that two resources should be called semiantically equivalent (e.g. "Two resources are semiantically equivalent if the labels of these two resources are at least 95% similar using Jaro Winkler similarity\(^3\)").

### 3.2.3 Rule-based inference

The description of semantic enrichment also touches on the topic of rule-based inference. Rule-based inference also covers part of the semantic enrichment definition, albeit implemented specifically using rules. W3C states that inference means "automatic procedures can generate new relationships based on the data and based on some additional information in the form of a vocabulary, e.g., a set of rules" [32]. These automatic procedures are often responsible for executing rules that state the particular new relationships to be added. Often an inference engine is used to execute a set of inference rules.

Let us now focus on the structure of inference rules. Inference rules often consist of two components, the antecedent and the consequent. The antecedent is the precondition that should be satisfied, while the consequent states the new relationship to be added if the antecedent is satisfied.

Most RDF storage engines like Sesame\(^4\) and Jena\(^5\) allow using inference engines, which in turn provide inference of OWL ontologies, RDFS schema, as well as plain inference of additional RDF triples based on the RDF data.

Two rule languages frequently used in the Semantic Web community are SWRL\(^6\) and RIF\(^7\).

---

\(^1\) [http://www4.wiwiss.fu-berlin.de/bizer/silk/](http://www4.wiwiss.fu-berlin.de/bizer/silk/)

\(^2\) [http://www4.wiwiss.fu-berlin.de/bizer/silk/spec/#conditions](http://www4.wiwiss.fu-berlin.de/bizer/silk/spec/#conditions)

\(^3\) [http://staffwww.dcs.shef.ac.uk/people/S.Chapman/stringmetrics.html#jarowinkler](http://staffwww.dcs.shef.ac.uk/people/S.Chapman/stringmetrics.html#jarowinkler)

\(^4\) [http://www.openrdf.org/doc/sesame2/users/ch08.html#d0e765](http://www.openrdf.org/doc/sesame2/users/ch08.html#d0e765)


\(^6\) [http://www.w3.org/Submission/SWRL/](http://www.w3.org/Submission/SWRL/)

3.3 Presentation of the enrichment process

Now that we have seen a number of topics related to semantic enrichment, we also discuss how semantic enrichment can be visualized. This section illustrates three frameworks, which use a similar environment for specifying data manipulation. The aim of this section is to provide an insight in their capabilities.

**Yahoo Pipes**  An environment that is specifically designed to represent an enrichment process is the Yahoo Pipes\(^8\) environment. The Yahoo Pipes environment allows the use and reuse of ‘pipes’ which are combined and connected. To add to this, when a series of pipes has been connected and the user starts running the pipe, a sequence of data fetching and data manipulation operations will be executed.

Yahoo Pipes does not aim particularly at the Semantic Web, but rather focuses on other types of data such as XML and JSON as well as RSS feeds. Yahoo Pipes is used a lot to mash-up data from different data sources [33].

**DERI Pipes**  The DERI Pipes\(^9\) environment is inspired on the Yahoo Pipes environment, in the sense that most of the user interface elements have been applied to DERI Pipes as well.

In contrary to Yahoo Pipes, DERI Pipes is designed to retrieve and manipulate RDF data [22]. Like in Yahoo Pipes, XML and JSON are supported.

The idea behind DERI Pipes allows the user to create mash-ups of Semantic Web data very quickly, by simply composing a number of operations [20].

**Taverna**  The Taverna\(^10\) system is a workflow management system in which workflows are composed of services[23].

![Figure 3.1: Example workflow in Taverna](http://wiki.cvrgrid.org/index.php/ECG_Analysis_Workflow)

---

\(^8\)http://pipes.yahoo.com
\(^9\)http://pipes.deri.org/
\(^10\)http://www.taverna.org.uk/
Figure 3.1 shows the visualization of a workflow in Taverna. It shows the data flow between the workflow input and output ports. Workflows can be literally dozens of services large, and can even include nested workflows. Workflows are used to conduct a series of operations, usually on a set of measurements. Executing the workflow will sequentially perform all operations that are necessary for performing the complex task.

Taverna is mostly used in the Life Science domain, so many services are applicable to this domain. Especially very specific operations in the Life Science domain, like operations concerning genetics are tackled by using these services and combine their output [17]. Many of the provided services make use of web services specified using the WSDL standard.

The capabilities provided by Taverna also include the reuse of services as well as complete workflows. A community has been formed around Taverna, which helps formulating more of these commonly reusable flows.

Extensibility also became a crucial point for Taverna, allowing the community to create completely new functionalities, and attach these to the Taverna workbench [23].

3.4 Requirements

This section will cover a set of requirements that were defined to specify the way in which RDF Gears supports semantic enrichment of RDF datasets, as well as how the user is guided through this process. Each of the requirements is explained in detail now.

3.4.1 Flows and blocks

The main paradigm of the semantic enrichment framework consists of flows and blocks. Flows are a declarative model consisting of a sequence of connected blocks, where each block stands for an operation that should be performed on the RDF data. The blocks in a flow are connected using input and output ports, which are associated to each of the blocks.

We intentionally decided not to choose inference rules as our main paradigm, because a set of inference rules (with potential entanglements) can become quite cumbersome to maintain. Also, flows and blocks are more interchangeable than a set of inference rules. Flows and blocks can be shared within communities, while a set of rules is more unlikely to be shared. The current implementation does not (yet) implement features for sharing and importing flows and blocks.

Note that it is not unlikely that flows and blocks are rewritten into a set of rules in the back-end of the RDF Gears framework. However, the presentation of the framework adheres strictly to flows and blocks.

3.4.2 Support for semantic data integration

In RDF Gears, we would like to be able to express mappings between a resource in the core RDF dataset and a resource in the auxiliary RDF dataset (i.e. the resource in the global schema) as defined by Cruz and Rajendran [12]. In order to be able to support these mappings, we introduce a mechanism called 'object typing'.

19
3.5 Conclusion

Semantic enrichment

An object type is a set of defined properties (with associated values), which is used to describe different types of resources in the global schema. If for any resource within the global schema (auxiliary dataset) the defined conditions hold, this resource is a member of this object type.

Object types may adhere to the explicit typing that is used within the auxiliary dataset, which is often defined with the \texttt{rdf:type} predicate. The Geonames dataset\footnote{http://www.geonames.org} is a clear example however where the object typing is necessary, because all resources are of the same \texttt{rdf:type}, namely \texttt{geonames:Feature}. In order to be able to distinguish resources representing a city, region, country or another geographical entity\footnote{for a complete list of entities, see: http://www.geonames.org/statistics/total.html}, the predicate \texttt{geonames:featureClass} should be used instead of the \texttt{rdf:type} predicate.

The object types are used for type checking during the enrichment process. Also, connections between blocks in the enrichment flow are validated based on the object types that are attached to the ports of a block.

During the specification of an enrichment flow, the object typing mechanism is used for checking the validity of the connections in the enrichment flow. Also, the object typing mechanism is used for recommending blocks that might be used next while specifying an enrichment flow. More uses may be found for the object typing mechanism in the future.

3.4.3 Support for semantic interlinking

For the implementation of the RDF Gears platform, we choose to support the semantic interlinking process by including the Silk interlinker tool into RDF Gears. If a Silk interlinking has been defined in the enrichment flow, the Silk interlinker tool will execute the Silk LSL file (interlinking file). After the execution of the Silk LSL file, RDF Gears is able to access the RDF data that was generated by Silk (i.e. the interlinkings), and to process this data in the next steps of the enrichment flow.

3.5 Conclusion

As was described in this chapter, semantic enrichment is the process of starting with an RDF dataset and incrementally adding more data to the resources which are described in this RDF dataset. Based on the values of the properties describing the resource, new knowledge can be inferred and added to the RDF dataset to make it richer.

The RDF Gears framework, that is developed during this Master’s Thesis Project, takes an approach to which any RDF dataset can be enriched by gathering RDF data from other RDF datasets that are publicly available. Obviously, depending on the nature of the dataset, possibilities for semantically enriching the initial RDF dataset will differ.

This chapter also discussed the close relation to semantic data integration, semantic interlinking and rule-based inference. RDF Gears aims at supporting the former two approaches, while the latter is not implemented by default. As was stated before, a
rule-based inference engine can be used as an alternative approach to the system of flows and blocks.

The presentation style chosen by Taverna, DERI Pipes and Yahoo Pipes is also adopted in RDF Gears, since it shows how data flows in an intuitive way.

Requirements for the semantic enrichment framework include that flows and blocks are used to allow the specification of the enrichments that are performed on the core RDF dataset. The framework also needs a mechanism to support semantic data integration. This mechanism is called ‘object typing’ and will be discussed in Section 4.2.3.

Part I has illustrated the technologies and concepts behind the Semantic Web and semantic enrichment (i.e. making RDF datasets richer). The remainder of this document will illustrate the solution that was built in order to create the semantic enrichment framework called RDF Gears.
Part II

RDF Gears
Chapter 4

High-level design of RDF Gears

Chapter 3 discussed the general process of enriching RDF data. In this chapter, the high-level design of RDF Gears will be discussed, starting by outlining the general process of enriching RDF data in the RDF Gears application. Afterwards, the models behind the semantic enrichment process itself are described in detail. The models are directly related to the basic conceptual process that will be described first.

4.1 Conceptual process of enriching RDF data in the RDF Gears framework

The basic conceptual process of enriching RDF data in RDF Gears can be characterized as a sequence of four main steps. The complete sequence of processes is shown in Figure 4.1. Each of these steps is discussed in more detail below.

Note that the steps depicted in Figure 4.1 are not applicable for all semantic enrichment frameworks, but rather depict the specific process used by the RDF Gears framework.
4.2 Models driving the semantic enrichment process

Import core RDF data  The process of enriching RDF data with RDF Gears starts with selecting an appropriate source of RDF data. The RDF data is imported from a file so that RDF Gears can enrich this RDF dataset. The imported RDF data will be stored in an RDF store.

Specify enrichment flow  The next step is specifying an enrichment flow (or loading a previously saved enrichment flow). The enrichment flow contains the specification of all operations that should be performed on the core RDF data, and how auxiliary RDF datasets are used to enrich the core RDF data. The enrichment flow model will be explained in more detail in Section 4.2.5.

Basically the user interacts with the canvas of the RDF Gears user interface, and constructs an executable enrichment flow model. The enrichment flow model specifies the RDF data from the core RDF dataset that should be enriched, as well as how these enrichments are stored in the enriched dataset. Enrichment flows can be saved for later use, so that the user does not have to recreate the entire enrichment flow.

Execute specified enrichment flow  When the enrichment flow has been specified, RDF Gears is able to execute the enrichment flow, and enrich the imported RDF data accordingly. The flow is decomposed into separate elements, which are then executed in the correct order. A detailed description of this process will be given in Section 4.3. After the enrichment flow has been executed, it is possible to specify a new enrichment flow.

Access enriched RDF data  After the enrichment flow has been executed by RDF Gears, the enriched RDF data can be accessed. Two different ways of accessing the enriched RDF data are (1) exporting the enriched RDF dataset to a file and (2) expose the enriched RDF data via a SPARQL endpoint\(^1\).

4.2 Models driving the semantic enrichment process

Since RDF Gears is designed as a model-driven approach to creating Semantic Web applications, the main models driving RDF Gears will now be explained. The essence of model-driven approaches is that models are used to generate and/or verify results. RDF Gears uses a total of 3 models, namely (1) the input model (Section 4.2.1), (2) the output model (Section 4.2.2) and (3) the enrichment flow model (Section 4.2.5).

In order to explain the enrichment flow model, the notion of a ‘block’ will be given in Section 4.2.3, which has a strong connection to the notion of ‘enrichment step’ (Section 4.2.4).

\(^1\)RDF Gears uses the Sesame RDF store, which provides a SPARQL endpoint over the RDF data
4.2.1 Input model

The input model describes the RDF data that will eventually be enriched. The input model expresses the data model of the imported RDF dataset, which is an RDF graph. A simple input model is shown in Figure 4.2. The RDF graph that is shown in Figure 4.2 displays resources as green ovals and literal data as oranges boxes. The labels near the directed edges depict RDF predicates.

4.2.2 Output model

The output model describes the RDF data that was imported (i.e. the input model) augmented with the triples that will be generated during the execution of the enrichment flow. Similarly to the input model, the output model is also expressed in the RDF
4.2 Models driving the semantic enrichment process

High-level design of RDF Gears

data model. Figure 4.3 shows an example output model, that looks similar to the input model depicted in Figure 4.2, but displays a new RDF triple\textsuperscript{2}.

4.2.3 Blocks

Blocks are executable subcomponents of enrichment flows, which will be discussed in Section 4.2.5. First, the basic structure of the block is outlined. Then we continue by discussing the three categories of blocks that are available in the block repository (i.e. the set of all blocks that can be placed in the enrichment flow).

Inputs/outputs With each block we associate a number of inputs and outputs. These inputs and outputs are called ‘ports’. A simple example of how a block will be displayed in the enrichment flow model has been shown in Figure 4.4\textsuperscript{3}.

![Figure 4.4: Example of a block representation in the UI](image)

As can be seen in Figure 4.4, a block has a number of input and output ports. These are displayed as labels annotated with symbols to indicate the type of port. When a circle is shown near the port, the port has no type. In other words, it would accept everything that either would come in or go out. However, whenever a square is shown, the port does have a type. Typed ports are used for object typing, which is discussed next. Note that the set of input ports as well as the set of output ports may contain both untyped and typed ports. Table 4.1 shows an overview of the port types and their representations.

![Untyped port](image), ![Typed port](image)

Table 4.1: Overview of port types and their representations in the UI

Object typing One of the requirements of the RDF Gears framework is that a mechanism called object typing is introduced. As was discussed in Chapter 3, it provides a way to create links from resources within one RDF dataset and a ‘global schema’. Object typing is used for (1) type checking during the enrichment process, (2) checking of connections (will be discussed shortly), (3) the recommendation of new blocks based on the blocks that are already part of the enrichment flow and (4) during the definition of new enrichment steps (see Section 4.2.4).

\textsuperscript{2}vcard:geo "<http://sws.geonames.org/3165243/>"
\textsuperscript{3}This image is taken from the RDF Gears user interface
One of the main issues we encountered is that the predicate `rdf:type` is not always a good measure to classify resources within a dataset. When looking at the Geonames ontology, one can see that all instances are of `rdf:type geonames:Feature`. Figure 4.5 shows two resources, which both have type `geonames:Feature`. Instead, one could use the link `geonames:featureCode` to classify resources on. The RDF Gears framework allows the creation of such object types. A resource is member of this particular object type if the resource has the predicate and value combination that is specified for the object type.

Figure 4.5 illustrates that the `rdf:type` cannot be used to create a classification. However with the object typing mechanism in place, object types can be created that can be used to properly classify resources in an RDF dataset. In this case, the object types ‘Geonames City’ and ‘Geonames Region’ have been made using the predicate and object depicted in Table 4.2. The distinction here is based on the value associated to the predicate `geonames:featureCode`, rather than on the value associated to the `rdf:type` predicate.

<table>
<thead>
<tr>
<th>Object type name</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geonames City</td>
<td>geonames:featureCode</td>
<td>geonames:P.PPLA</td>
</tr>
<tr>
<td>Geonames Region</td>
<td>geonames:featureCode</td>
<td>geonames:A.ADM1</td>
</tr>
</tbody>
</table>

Table 4.2: Overview of the object types depicted in Figure 4.5

Each object type is uniquely identifiable using a URI. RDF Gears uses these object types to performed named typing (i.e. the URI of the expected object type should match the URI of the actual object type), which is especially used for checking the connections within an enrichment flow (see Section 4.2.5). No type conversion takes place when checking whether object types are equivalent.

We briefly introduced that object typing is used when performing SPARQL queries. When executing a SPARQL query, we can filter the results by checking whether the resources that were found using a SPARQL query (in an auxiliary RDF dataset) satisfy
the criteria for being member of the particular object type. If the conditions do not satisfy, this resource will be omitted.

**Block categories** There are three categories of blocks that are stored in the block repository. Each of these types can occur in the enrichment flow model:

- The first category contains the 'input predicates block' and 'output predicates block', which control the connections between the input and output model and the remainder of the blocks in the enrichment flow. Both of these blocks occur exactly once in each enrichment flow.

- The second category are so-called 'enrichment steps'. These are basically the steps that manipulate RDF data and therefore determine the rules for creating enriched RDF data. Enrichment steps will be explained shortly in Section 4.2.4.

- The third category contains a set of blocks with a pre-defined meaning. There are 5 special operations in this category:
  - the 'datatype block', this block allows the user to gather datatypes from resources within the auxiliary dataset;
  - the 'filter block', this block allows the user to restrict the resources to which to apply enrichment (e.g. "only stores located in the city of Delft");
  - the 'matcher block', this block allows to convert an untyped input to a (verified) typed output\(^4\);
  - the 'Silk block', this block allows the user to execute a Silk LSL-file. Silk is able to interlink the contents of the imported RDF dataset with an auxiliary RDF dataset;
  - the 'union block', this block is used to combine the values of two incoming connections into one set of values.

### 4.2.4 Enrichment steps

When the block is of the second category, this block is known as an enrichment step block. An enrichment step is basically the function between input(s) and output(s). The definition of the enrichment step prescribes the action to be taken when this enrichment step is being executed.

Enrichment steps are building blocks for creating semantic enrichments, since they allow the manipulation of RDF data. Each enrichment step has a URI, which allows the enrichment step (and thus its function) to be retrieved from the block repository using this URI.

In the current implementation, the enrichment step can be of 4 different kinds, each of which can be used in a different scenario.

**SPARQL enrichment step** The SPARQL enrichment steps are bound to a specific SPARQL endpoint, and can perform SPARQL SELECT queries. Often SPARQL queries can be used to traverse RDF links within auxiliary RDF datasets.

\(^4\)More details about object typing can be found in the previous paragraph
Java enrichment step  Java enrichment steps can be used to perform both semantic as well as non-semantic operations (such as arithmetics), the latter being not fully supported in SPARQL. Cryptography functions like MD5, SHA1, and string similarity algorithms like Jaro Winkler and Levenshtein are part of the RDF Gears framework as Java enrichment steps.

API enrichment step  An Application Programmer Interface (API) can be called from this enrichment step type. Two API enrichment steps are packaged in the RDF Gears distribution, the first uses the Geonames Search service, the second uses the DBpedia Look-up service. Essentially the API enrichment step is a special case of a Java enrichment step, since both use procedural Java code to manipulate data. The API enrichment step was introduced because in the future one might be interested to classify enrichment steps based on their nature. Usually API enrichment steps are used to retrieve XML or RDF data using the HTTP protocol.

Compound enrichment step  Finally, compound enrichment steps are available to store a fully specified enrichment flow as a single enrichment step. Executing a compound enrichment step will result in executing all blocks within the nested enrichment flow. Afterwards the execution of the specified (top-level) enrichment flow will proceed.

Next to these predefined enrichment step types, the RDF Gears framework can be extended with other types of enrichment steps. By extending the number of enrichment step types, it is possible to create enrichment steps, that manipulate data using another paradigm (e.g. using the Python programming language or the SPARQL 1.1 Query language).

4.2.5 Enrichment flow model

The enrichment flow model is the core concept of RDF Gears. It specifies the operations that should be performed on the core RDF dataset (i.e. how RDF data needs to be manipulated in order to infer new data). An enrichment flow is defined as a sequence of blocks (see Section 4.2.3). These blocks are connected via directed edges. Each block stands for an operation that should be performed on the RDF data. The enrichment flow model is a declarative model that specifies how the core RDF dataset should be enriched, while allowing the back-end of the RDF Gears framework to be interchangeable. An example of an enrichment flow model is shown in Figure 4.6.

---

6 http://tools.ietf.org/html/rfc3174
7 http://staffwww.dcs.shef.ac.uk/people/S.Chapman/stringmetrics.html#jarowinkler
8 http://staffwww.dcs.shef.ac.uk/people/S.Chapman/stringmetrics.html#Levenshtein
9 http://www.geonames.org/export/geonames-search.html
10 http://lookup.dbpedia.org/
11 While Java enrichment steps should be used for manipulating data using only Java code
12 http://www.python.org/
13 http://www.w3.org/TR/sparql11-query/
4.2 Models driving the semantic enrichment process

High-level design of RDF Gears

Semantics  The semantics of the enrichment flow are as follows:

The enrichment flow contains a set of blocks (with input and output ports) and connections between blocks. Enrichment flows specify all operations that should be performed on the RDF data in a declarative way. The semantics of a block follow from the block type, and if applicable the attached function, while the semantics of the edges between blocks specify the execution dependency between blocks (e.g. "block x should be executed before block y"). Enrichment flows contain two special blocks, which specify how the input model is interpreted (i.e. the 'Input predicates' block) and how the enriched RDF data will become part of the output model (i.e. the 'Output predicates' block).

Syntax  The syntax of the enrichment flow is as follows:

The complete enrichment flow is a function that maps one RDF graph to a richer RDF graph. Each of the blocks within the enrichment flow projects one set of RDF predicates for a particular RDF resource to a new set of RDF predicates for the same RDF resource.

The ports of blocks may only be connected if the type of the ports (i.e. the object types) are the same. Name-based checking is used to verify whether two object types are equivalent.

An executable enrichment flow is an acyclic graph. This property is necessary for being able to execute enrichment flows, because a cycle in the enrichment flow would mean that neither block can be executed before the other.

Connections are directed edges between an output port and an input port. Any output port can have multiple outgoing edges, but any input port can only have one incoming edge. The types for these ports need to correspond properly (i.e. whenever both ports are typed, the object type needs to correspond).

While importing a core RDF dataset into RDF Gears, all RDF predicates from this dataset will be retrieved and displayed in the 'Input predicates' block.

The RDF Gears framework provides a syntax-based editor, which checks whether (1) all edges between blocks are correct and (2) no cycles occur in the specified enrichment flow.

Implementation  The enrichment flow is a connected directed acyclic graph (DAG). Directed acyclic graphs are graphs with directed edges between the nodes, and no directed cycles. When the enrichment flow is encoded as a DAG, we can apply partial

![Enrichment flow model](image)

Figure 4.6: Example of an enrichment flow model
order sort to find a proper execution order. The partial order sort is a sorting technique that can be applied to a graph, which in this case is a dependency graph. The partial order sort algorithm creates a proper execution order by stating that node $x$ must be executed before node $y$ if $x$ has an outgoing edge to node $y$. Figure 4.7 shows a DAG with a valid execution order. Note that a total of 5 execution orders are correct.\footnote{12345, 12435, 21345, 21435 and 24135}

![Figure 4.7: Example of DAG and a valid execution order](image)

The enrichment flow that was introduced in this section contained only one enrichment step, but the RDF Gears framework allows the creation of significantly larger enrichment flows. Figure 4.8\footnote{This image is taken from the RDF Gears user interface} shows an enrichment flow containing more than just one block for enriching RDF data. This particular case combines two RDF predicates from the core RDF dataset\footnote{swrc:address and swrc:location}, and creates one new RDF predicate in the enriched RDF dataset\footnote{foaf:based_near} (i.e. the output model).

![Figure 4.8: Example of an enrichment flow containing more blocks](image)

### 4.3 Executing the enrichment flow model

Previously, we have discussed the models behind the semantic enrichment process. This section elaborates on the execution of the enrichment flow model. This process involves a sequence of steps, which is exemplified by composing the model examples that have been presented earlier this chapter. We show how an example input model and enrichment flow are converted to an output model in Figure 4.9.

\footnote{12345, 12435, 21345, 21435 and 24135\footnote{This image is taken from the RDF Gears user interface}\footnote{swrc:address and swrc:location}\footnote{foaf:based_near}
4.3 Executing the enrichment flow model

The intention of the enrichment process is creating the output model as was described earlier. In general, the output model consists of the contents of the input model, and additional RDF triples, that have been found while executing the enrichment flow. We will now discuss the three generalized steps in detail, according to the example shown in Figure 4.9.

Our first assumption is that we can store both our input model and output model into the same RDF graph. Notice that the example shown in Figure 4.9 displays a simplified situation, in which input model only contains one resource. The enrichment flow also might have contained more than one block, as was discussed in Section 4.2.5. For practical reasons, a simple example is shown to illustrate the idea behind the enrichment flow execution.

**Action 1: Copy data from input model to output model**  When the enrichment flow execution starts, the RDF data in the input model is copied to the output model (i.e. each RDF triple in the input model is now also represented in the output model).

**Action 2: Execute enrichment flow**  Now the enrichment flow is executed. First, the execution order is determined (by applying partial order sort to the enrichment flow, which is in essence a DAG). Then, the blocks specified in the enrichment flow are executed sequentially, according to the execution order.
For each block, we iterate over its input ports. Each input port is associated to an RDF predicate, which is used to store the values that enter the port. All resources with these predicates are now retrieved from the RDF dataset, after which the block operation is being applied to the values associated to these predicates. For each output port, we generate a new RDF predicate, and create RDF triples linking each resource to the result value of the operation (with the generated predicate). Note that whenever an output port is connected to an input port, they share the same RDF predicate (i.e. data stored in a previous block can thus be accessed in the current block by retrieving the values associated to the predicate).

In this case the input model contains only one resource. The 'Geonames City lookup' block receives this resource and the value "Trento, Italy". The 'Geonames City lookup' block finds the value http://sws.geonames.org/3165243/, and associates this to the same resource, with a newly generated predicate.

**Action 3: Copy new triples to output model**  Now the new triples found in the enrichment flow execution are copied to the output model. In this (simplified) case the following RDF triple is added:

\[
_:A\_Framework\_for\_Flexible\_User\_Profile\_Mashups \quad \text{vcard:geo} \\
<http://sws.geonames.org/3165243/>
\]

Note that the gray box depicted in Figure 4.9 shows that the correct resource within the Geonames dataset has indeed been found by the 'Geonames City lookup' block.

It should be stressed that this process shows a simplified enrichment flow model. If the input model contains more than one RDF resource, each block is executed for each separate resource, before the execution of the next block in the enrichment flow model is started.

While the input and output model are conceptually different, we physically store them in the same RDF graph. It is possible to use separate RDF graphs to store the input and output model, but this choice would involve the need to copy RDF data from one graph to the other. Choosing for just one RDF graph to store both models saves both time and space, because the RDF data does not have to be duplicated.

### 4.4 Conclusion

In summary, the main process for enriching RDF data is a sequence consisting of the following 4 steps:

- Import core RDF data;
- Specify enrichment flow;
- Execute specified enrichment flow;
- Access enriched RDF data.

\[^{18}\text{_:A\_Framework\_for\_Flexible\_User\_Profile\_Mashups}\]
The enrichment flow model connects the input model (or core RDF dataset) to the output model (enriched RDF dataset). The enrichment flow describes which predicates from the input model are chosen to manipulate, similarly it also describes the predicates that should be added in the output model.

Blocks are available in three categories, namely (1) the special 'input predicates' and 'output predicates' block, representing the connection with the input and output model, (2) the 'enrichment steps'\(^\text{19}\), allowing RDF data to be manipulated using a SPARQL SELECT query, Java code or even (nested) sub-flows and (3) a number of blocks with a pre-defined meaning, such as the filter block, matcher block and union block.

Each enrichment flow contains a number of blocks, which are connected into a directed acyclic graph (DAG). When executing an enrichment flow, partial order sort is performed on the DAG. The result is a list of nodes in proper execution order.

The execution of the enrichment flow takes place in a number of steps. These steps are summarized now:

1. Copy data from input model to output model (at start of the execution);
2. Execute all blocks in the enrichment flow sequentially, based on the input that is available in the enrichment flow;
3. Create new triples in the output model, based on the predicates in the output predicates block.

\(^{19}\)The RDF Gears framework allows creating new enrichment steps of each type, as well as new enrichment step types, that could include future Semantic Web technologies.
Chapter 5

Architecture of RDF Gears

In Chapter 4, the models driving the semantic enrichment process have been discussed. This chapter takes the design and describes the implementation of RDF Gears and the basic components of the RDF Gears framework. We first introduce the architecture of the RDF Gears framework.

Figure 5.1 shows the RDF Gears framework consists of 4 main components. Each of the main components is discussed separately now, starting with the RDF storage component.

5.1 Storage of RDF data

The RDF Gears platform uses a dedicated RDF store. In the current implementation, the server version of Sesame¹ is chosen as the RDF store. Instead of using Sesame, one could also adapt the code quite easily to connect to another RDF store, like Jena² or Virtuoso³.

The RDF storage component within the RDF Gears framework is responsible for storing the core RDF dataset in an RDF graph. When enriching the imported RDF dataset with RDF Gears, the newly generated RDF triples will be added to the RDF

¹http://www.openrdf.org/about.jsp
²http://jena.sourceforge.net/
³http://virtuoso.openlinksw.com/
graph that also stores the core RDF dataset. The RDF storage component also allows exporting the full RDF dataset to a file.

5.2 Core classes

The RDF Gears system has a number of core classes, which are used in the later discussed 'user interface' and 'enrichment engine' components. This set of classes defines the models identified in Chapter 4.2 in terms of Java classes. This allows RDF Gears to perform the enrichment of RDF data as described in Chapter 4. Let us continue by outlining the basic classes of the RDF Gears core.

Flow  The Flow object corresponds to the enrichment flow model which was discussed in Section 4.2.5. The Flow object is the data structure that describes an enrichment flow, by specifying its blocks and connections. The Flow class does not prescribe any implementation for the back-end, which will be described in detail in Section 5.4.

In order to allow saving and loading of enrichment flows in the RDF Gears user interface, classes to store the Flow into an XML format are provided. The enrichment step models that are encapsulated in the enrichment flow are stored separately, as will be discussed shortly.

Block  The Flow object describes the blocks and connections within an enrichment flow. The abstract class Block reflects the notion of 'block' that was introduced in Section 4.2.3. Implementations for each of the block types as described in Section 4.2.3 are also provided in the core package.

EnrichmentStep  The enrichment step model identified in Chapter 4 is covered by the abstract EnrichmentStep class. Implementations for each of the 4 sub-types (SPARQL enrichment step, API enrichment step, Java enrichment step and compound enrichment step) are part of the core package. Instances of all EnrichmentStep implementations hold a URI parameter in order to identify the unique enrichment step. This is used by the RDF Gears framework to retrieve the function associated to the enrichment step by only using the URI.

RDF Gears uses a controller class called EnrichmentStepController, which contains all enrichment steps that were read from the enrichment step files. Flows only store the URI of the enrichment step whenever an enrichment step is part of an enrichment flow. The EnrichmentStepController can be requested an enrichment step (and thus its function) by looking for a URI.

Like the Flow class, the implementations of the EnrichmentStep class have capabilities to save and load. The reason for this is that specific enrichment steps will be reused in several enrichment flows. Also, separating enrichment steps from the enrichment flow models accommodates exchanging enrichment steps between users.

The RDF Gears framework also provides a platform for creating new enrichment steps. This functionality is mainly provided by the wizards in the user interface.

Dataset  RDF Gears allows the use of a number of auxiliary datasets, which are used to enrich imported RDF data. The RDF Gears framework needs to have the proper
information for retrieving data from auxiliary RDF datasets (e.g. the location of a SPARQL endpoint for this dataset).

The Dataset class holds a number of parameters including the name of the dataset and an icon that can be shown in the user interface. Also, the URL of the SPARQL endpoint used for enrichment with the SPARQLEnrichmentStep is defined in instances of the Dataset class.

Object Type The object type mechanism (see Section 4.2.3) relies on the Dataset class that was described above. Instances of the class ObjectType mainly hold a display name and the criteria that should be satisfied in order to check whether a resource is member of this object type. These criteria are predicate-object pairs, which means that the resource should have a specified predicate with the corresponding value associated to it. Object types are specified on a particular auxiliary dataset (i.e. a specific Dataset instance).

5.3 User interface

The user interface of RDF Gears is built on the libraries SWT\textsuperscript{4} and Draw2D\textsuperscript{5}. SWT allows creating cross-platform user interfaces, while Draw2D is used primarily to let the user interact with the drawing canvas. This section will discuss categories of classes that were necessary to create the user interface of RDF Gears. The appearance of the user interface will be described in detail later in this section.

Canvas The main feature of the RDF Gears user interface is a canvas, which will eventually contain blocks and connections. The canvas functionality is part of the Draw2D library, in particular the class org.eclipse.draw2d.FigureCanvas. A FigureCanvas object stores objects of type org.eclipse.draw2d.Figure, which in the case of our user interface are blocks and their connections.

Tabs In order to provide parameters to blocks, tabs were introduced. When a block shown in the canvas is selected, a tab is opened displaying the options to alter the behaviour of the selected block during execution of the enrichment flow. Tabs are managed by the user interface using the TabController class.

Recommendation of blocks While specifying enrichment flows, one can go astray in the number of enrichment steps and other blocks that are available for specifying enrichment flows. The RecommendationController class is used to generate recommendations. These recommendations encompass the potential next steps while specifying an enrichment flow with RDF Gears. At this moment a simple recommender is available, which is discussed next\textsuperscript{6}.

\textsuperscript{4}http://www.eclipse.org/swt/
\textsuperscript{5}http://www.eclipse.org/gef/draw2d/index.php
\textsuperscript{6}Other recommendation techniques are possible to be added in the future, such as recommenders using data mining and pattern analysis on enrichment steps.
5.3 User interface

When a block is selected, recommendations will be generated by looking in the block repository (i.e. catalog containing all enrichment steps for potential predecessors) and successors (i.e. blocks that can be placed after the selected block). This recommendation method recommends candidate enrichment steps by comparing port names and object types of all ports in the block to all available enrichment steps in the block repository. If a match has been found, either based on type, or because the port name looks reasonably similar, the enrichment step will be nominated as a candidate enrichment step.

Recommendations will be created for the complete flow if no blocks are selected, or when the selected block is too general to create recommendations based on this block (e.g. the filter block or the union block). This process repeats the block recommendation process just described for each block in the enrichment flow. The recommendations will not be merged, but instead shown as a list of per-block recommendations.

Wizards RDF Gears is also a platform for creating user content. In general, one can create new enrichment steps, datasets, object types (discussed in Section 5.2), but also enrichment engines (see Section 5.4) and new enrichment step types.

Wizards are provided by RDF Gears to facilitate this process. Each of these wizards will guide the content creator through a series of operations. Some of the wizards require adding Java code to the RDF Gears framework in order to make the new functionality part of the RDF Gears system.

Appearance of RDF Gears user interface This component is responsible for displaying the graphical user interface which is the environment that allows the specification and execution of enrichment flows. A basic overview of the GUI is shown in Figure 5.2.

Figure 5.2: Overview of the RDF Gears GUI
5.4 Enrichment engine

An enrichment engine is designed to enrich the imported RDF data by executing the enrichment flow that was constructed in or loaded into the user interface. The enrichment engine receives the proper execution order of the enrichment flow, as discussed in Section 4.2.5. The enrichment engine is covered by the abstract class Engine.

The enrichment engine (i.e. an implementation of the abstract class Engine) can trigger the execution of the enrichment flow by instantiating an EngineRunner. This class takes the execution order of the enrichment flow, and for each encountered enrichment step the appropriate execution component will be run\(^7\). This process is likely to loop, since enrichment flows can contain more than one enrichment step type. A sequence diagram of the enrichment engine and related classes is shown in Figure 5.3.

![Sequence diagram overview of enrichment engine](image)

Each enrichment engine has a number of execution components of various types associated to it. These execution components have the task to perform the required action, which is retrieving and manipulating the RDF data and inserting the resulting RDF links and values in the RDF storage. They are usually designed to execute a single enrichment step type, such as a SPARQL enrichment step or a Java enrichment step.

---

\(^7\)The basic behaviour of blocks that are not enrichment steps is also part of the EngineRunner class
The RDF Gears framework currently has a number of execution components, which are shown in Figure 5.4. A summary of their functionality is listed below:

**ARQ_SPARQLComponent**  This execution component is associated with objects of type `SPARQLEnrichmentStep`. If this component is activated, the ARQ library is used to send SPARQL queries to remote SPARQL endpoints.

**DS_SPARQLComponent**  The DS_SPARQLComponent is also associated with the `SPARQLEnrichmentStep`. The DistributedSAIL library is used to perform federated SPARQL queries [27, 34]. DistributedSAIL is a plug-in for Sesame Server. Its task is to answer queries on both the local SPARQL endpoint as well as (multiple) remote SPARQL endpoints. This component relies on the Sesame Server. If another RDF store is used instead, it is not possible to use the DistributedSAIL engine.

**CommonJavaComponent**  In order to execute `JavaEnrichmentStep` instances, the CommonJavaComponent was introduced. By default, this component executes all Java enrichment steps.

**CommonAPIComponent**  The CommonAPIComponent is used to execute API enrichment steps. In the default engines delivered with the RDF Gears framework, all enrichment steps of type `APIEnrichmentStep` are executed with this component.

**SilkComponent**  The final execution component is the SilkComponent. This component runs the Silk application from within the RDF Gears framework. After Silk has executed the Silk LSL-file, RDF Gears will be able to use these new interlinkings.
5.5 Conclusion

This chapter has identified the main architecture of RDF Gears. The 4 main components have been introduced.

The first component is the RDF Storage component. In RDF Gears, we assume that the Sesame Server is used as the primary RDF store. It is possible to use another RDF store instead, such as Jena or Virtuoso.

The second component contains the core classes of the RDF Gears framework. These classes represent the models as described in Chapter 4.

The third component is the user interface of the RDF Gears framework. This component is responsible for creating a suitable environment for the user, to allow the user to make enrichment flows. Also, the user interface provides wizards to create new content for RDF Gears.

The fourth component is the enrichment engine, which executes the enrichment flow specified using the user interface. Multiple enrichment engines can be used interchangeably to execute the enrichment flow.
Chapter 6

Evaluation of RDF Gears

Chapter 5 described the overall architecture of the RDF Gears framework. This chapter will evaluate the approach chosen for RDF Gears. We will compare RDF Gears with the methodologies which were introduced in Chapter 3. Also, we comment on the extensibility of RDF Gears. This chapter ends with a description of the expressivity of the visual enrichment specification language.

6.1 Comparing with other methodologies

This section will discuss the coverage of the topics related to semantic enrichment as discussed in Chapter 3. First we evaluate the coverage of semantic data integration in the RDF Gears platform. We continue with evaluating the coverage of semantic data interlinking and rule-based inference in the RDF Gears framework.

6.1.1 Semantic data integration

We described in Section 3.2.2 a total of 5 mappings identified by Cruz and Rajendran [12]. These mappings are (1) one-to-one, (2) one-to-null, (3) parent-children, (4) one-to-many and (5) many-to-one, where the mappings are defined from the global schema entities (i.e. entities in the auxiliary dataset) to the application data RDF entities. In the RDF Gears framework, each of these mappings is expressible using an enrichment step.

Using the object type mechanism, we are able to express resources in the global schema. The 5 mappings that were just discussed are performed by the enrichment steps, which in our case means that globally defined resources are extracted and made part of the local RDF dataset. The RDF Gears framework adopts the LA V (Local as a view) approach, which makes sure that any new auxiliary RDF dataset can be added to the RDF Gears framework without the need for changing the schema of the local datasources.

The system as described by Cruz and Rajendran focuses on integration of multiple global schema without user intervention, by providing a system that rewrites user queries [12]. However, instead of providing a query rewriting solution, the RDF Gears framework extracts information from the global schema (i.e. auxiliary dataset), and makes it part of the enriched RDF dataset. User intervention is necessary to spec-
ify how the extracted information should become materialized in the enriched RDF dataset.

6.1.2 Semantic data interlinking

As discussed in Section 3.4.3, RDF Gears does not have semantic interlinking capabilities itself. Instead, it uses the Silk interlinker tool, which can be applied to the imported RDF dataset (by using the SPARQL endpoint provided by Sesame) and any remote SPARQL endpoint.

Silk interlinkings may take considerable time to execute, because the interlinking process requires the retrieval of all data that should be compared. Afterwards, pairwise comparison is executed between data from both SPARQL endpoints. Silk 2 offers considerably quicker execution (a speed-up factor 20, compared to the original Silk release [18]) by reducing the number of comparisons immensely. At the time of writing, a new blocking method has also been proposed [18].

In Section 6.3 we compare the expressivity of the RDF Gears enrichment specification language with the Silk Link Specification Language (LSL).

6.1.3 Rule-based inference

Rule-based inference is not covered by RDF Gears, although the platform allows creating a rule-based inference engine. We suspect that most of the components of an enrichment flow can be rewritten into a set of inference rules which are equivalent. Most rule-based systems include a knowledge base (containing all facts) and a rule base (containing all rules). The RDF store can be regarded as a knowledge base, containing all true facts encoded in RDF triples. Inference rules then operate on the current state of the knowledge base, and generate new facts (i.e. new RDF triples).

As mentioned earlier, rule bases are often quite complex and not easily maintainable, however if the rule base is not visible for the user it is possible to use a rule-based enrichment engine. This approach implies that blocks within enrichment flows are translated into a set of inference rules.

6.2 Extensibility of the RDF Gears framework

We have shown that it is possible to include third-party systems like the Silk interlinker into RDF Gears. Depending on the nature of the system that is a candidate for including in RDF Gears, integration of the system will have to be determined. Some systems may be trivial to include, while others can be inherently difficult.

Some functionality obviously has to be developed, such as the interface between the third-party system and RDF Gears. Silk offers a command-line interface, which can be invoked from within RDF Gears. After Silk has finished executing a specific interlinking file (Silk LSL file), the new links are stored into a file that can be retrieved from the Silk LSL file. RDF Gears then accesses this file, and makes the new links part of the enriched RDF dataset. Including this behaviour in the RDF Gears framework required the SilkComponent class described in Section 5.4.

Two components that should be added to the user interface are a 'block' and a 'tab' implementation. The block is used to include execution of a third-party system in an
enrichment flow, while a tab displays the user the main parameters that are necessary for executing the system. These can be manipulated by the user. The execution component which is responsible for executing the included system will use these parameters.

6.3 Expressivity of the visual RDF enrichment language

The expressivity of the visual enrichment specification language follows from the syntax of the enrichment flow, as was described in Section 4.2.5. In that section we described that the enrichment flow is executed per resource within the RDF graph, but we will now elaborate on the types of operations that exist, and which of them are supported by RDF Gears. Afterwards we will compare RDF Gears to the Silk interlinker tool, to show that the RDF Gears visual enrichment specification language can be used to express a subset of operations of Silk.

Types of operations

Let us first introduce two major operation types that are available, namely (1) resource-based operations and (2) graph-based operations.

Resource-based operations are operations that are applied to one particular RDF resource at a time and all the predicates and objects that are associated to it. In this document, we have shown examples of the resource-based approach, in particular in Section 4.3. In particular, the RDF Gears framework is designed to perform resource-based operations. The basic execution models are designed to suit the resource-based approach. This means that whenever an enrichment step is executed, RDF Gears retrieves the RDF resources and their properties. The results of the data manipulation operation (e.g. the SPARQL SELECT query) are stored as properties of the selected RDF resource.

Graph-based operations involve operations on RDF graphs. One example of such an operation is the merge operation, which takes two RDF graphs, and outputs only one graph, which combines all data from both separate graphs. Graph-based operations often change the structure of a graph. RDF Gears is not suited for performing graph-based operations such as merging RDF graphs, because basic operations take place at the level of RDF resources. One could easily see that ontology matching belongs to the category of graph-based operations, since it tries to change the graphs representing two ontologies in order to find corresponding resources.

Expressivity of Silk operations in RDF Gears

We first present an example provided with the Silk interlinker tool. The example shows that resources describing movie directors in the DBpedia dataset can be interlinked with a resource in the LinkedMDB dataset that corresponds to this director. We slightly modified the LSL-file to look only for exactly matching strings as shown in Listing 6.1.

---

Original example: http://www4.wiwiss.fu-berlin.de/bizer/silk/spec/#DBpedia2LinkedMDBdirector

Original example: http://www4.wiwiss.fu-berlin.de/bizer/silk/spec/?DBpedia2LinkedMDBdirector

Original example: http://www.dbpedia.org/

Original example: http://www.linkedmdb.org/
Listing 6.1: adapted Silk mapping file
In RDF Gears, we can also express this interlinking. We assume the core RDF dataset contains a list of movies and their directors in the DBpedia dataset, and create an enrichment flow to store resources representing the director in the LinkedMDB dataset using the dbo:director predicate. The enrichment flow is shown in Figure 6.1.

![Figure 6.1: Enrichment flow that is equivalent to the adapted Silk mapping](image)

The 'Find director' block is described by the SPARQL query shown in Listing 6.2.

```sparql
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>

SELECT ?director
WHERE {
    {?director rdf:type movie:director .
     OPTIONAL {?director movie:director_name ?l1 .
             FILTER(?l1 = str(?label))}
    UNION{
     {?director rdf:type movie:director .
      FILTER(?l2 = str(?label))}
    }}
}
```

Listing 6.2: SPARQL query attached to the 'Find directors’ block

From this simple example we can see that the visual enrichment flow specification language of RDF Gears is less verbose compared to the specification language used by Silk. The Silk interlinker tool does not provide a syntax editor, so all LSL files have to be written by the user.

The RDF Gears framework can be used to perform a subset of the operations that Silk provides. This subset of operations can be expressed visually, which does not require the skills and knowledge to create a fully specified LSL file as shown in Listing 6.1.

### 6.4 Conclusion

In this chapter we have shown that RDF Gears covers semantic data integration mappings. Semantic data interlinking is not performed by RDF Gears itself, but RDF Gears allows the Silk interlinker to perform this job while executing an enrichment
flow. Finally we showed that RDF Gears is not a rule-based inference system, but it is possible to rewrite components in an enrichment flow into a set of inference rules.

Next we showed the potential for including third-party applications in the RDF Gears framework (i.e. executing the third-party application while enriching RDF data). We mentioned that RDF Gears uses the Silk interlinker tool, and that it is possible to include other third-party frameworks or applications in the RDF Gears framework.

We illustrated that RDF Gears is primarily used to perform resource-based operations. The expressivity focuses on these resource-based operations. Graph-based operations are not easily expressible in the RDF Gears visual enrichment flow language. Also we showed that the RDF Gears framework is able to express a subset of Silk operations. This subset is expressed using the RDF Gears visual enrichment flow language, which is less verbose than the Silk Link Specification Language.

We think that RDF Gears provides an interesting set of operations. Allowing the visual composition of enrichment steps opens up semantic enrichment to non-programmers. Our guess is that blocks and flows are likely to become interchangeable in the future, allowing complex operations to be reused within the community. We are convinced that RDF Gears provides the tools that are necessary to support such a user community system.
Part III

Conclusions and future work
Chapter 7

Conclusions and Future Work

This chapter outlines the conclusions regarding the research goal. First we illustrate the main conclusions regarding our research goal and whether we achieved this goal. Then we summarize the answers given on each of the research questions. The contributions we made during this Master’s Thesis project are described. This chapter concludes with an overview of identified future work.

7.1 Conclusions

This section reviews our research goal, and determines whether or not this goal has been reached. The research goal of this thesis project was defined as:

"Investigate and construct a model-driven environment that allows enriching RDF data in an intuitive way, by adding additional information from external sources and without posing any substantial restrictions on the source RDF dataset."

In order to achieve this goal, first we needed to create a thorough understanding of Semantic Web technologies, in particular RDF and SPARQL. These two technologies provided the basis upon which semantic data enrichment is performed.

Semantic data enrichment was discussed next. Although in literature there is no clear distinction between semantic data enrichment and two other commonly used terms (semantic data integration and semantic interlinking), we briefly described each of those techniques.

We then designed a set of models, which together guide the process of enriching RDF datasets. Having these models allowed to form an architecture, which describes the main components of RDF Gears. The implementation of the RDF Gears framework follows from this architecture.

The RDF Gears framework shows how semantic enrichment can be applied by specifying an enrichment flow, consisting of blocks, and consecutively being able to execute this enrichment flow.

We investigated the RDF Gears framework by comparing it to other techniques and products, and have established an understanding of the operations that can be properly expressed in the RDF Gears framework. Resource-based operations are properly expressible, while set-based and graph-based operations are not that well expressible
7.2 Summary per research question

The research goal of the Thesis Project was addressed by identifying and answering a number of key research questions. Each of them will be discussed below.

What is the Semantic Web and which standards are used to create Semantic Web applications? In Section 2.1, we showed that the Semantic Web is a web in which machines are used to assist humans in performing tasks such as searching. The primary way of doing so is by introducing the Web of Data, which adds meaning to the Web of Documents (the regular World Wide Web). RDF and SPARQL are two key standards which are used while creating Semantic Web applications.

What is Linked Data and what are the benefits of using Linked Data? We identified in Section 2.2 that Linked Data consists of a set of principles which should be used as guidelines while publishing public data. In particular, it prescribes the use of the RDF data model as the data format in which data should be accessible. Linking data to the Linked Data cloud provides better browseability and navigation of the linked dataset.

What is semantic enrichment, and how can the process of semantic enrichment be facilitated? Semantic enrichment has been described in Chapter 3 as a process in which new knowledge is added to resources in a dataset, by using the properties that currently describe the resources. The presentation of the semantic enrichment process in the RDF Gears platform resembles the visual programming paradigm used by Taverna.

What is the basic conceptual process of the RDF Gears framework to guide semantic enrichment? In Section 4.1 we identified that the semantic enrichment process in the RDF Gears framework can be described in 4 steps: importing an RDF dataset, specifying an enrichment flow, executing the enrichment flow and finally accessing the enriched RDF dataset.

Which are the models that drive the process of creating a semantically enriched RDF dataset? We identified three main models in Section 4.2. First we identified the input and output model, which express RDF data in the RDF data model. We further identified the enrichment flow model (a model specifying blocks and their connections). Blocks represent operations, while some blocks contain an enrichment step, which is the fundamental building block to transform the RDF data.
What is the architecture used by the RDF Gears framework to enrich RDF data?
RDF Gears is built from 4 components, as is described in Chapter 5. Each of these components has become a Java package. We identified the Storage component, which allows importing and exporting of RDF data into an RDF store, as well as querying RDF data. The next component consists of core classes, which represents the models described in Section 4.2. On top of the core classes component, we find both the user interface and the enrichment engine (i.e. the interchangeable back-end).

How does the RDF Gears framework relate to other systems, technologies and approaches?
In Chapter 6 we have seen how RDF Gears performs in term of expressivity, when compared to the semantic interlinking tool Silk as well as to other methodologies. RDF Gears shows that an interesting set of enrichment operations can be performed. The RDF Gears framework mainly focuses on resource-based operations. We concluded in Chapter 6 that RDF Gears is able to express a subset of operations of Silk, using a less verbose syntax. RDF Gears is a platform suited to perform resource-based operations by design.

When looking back at our research goal, we are convinced that the research goal has been achieved. We think that the research questions formulated to bring us to our goal have been answered properly. Semantically enriching an RDF dataset is a non-trivial problem, which required a specific set of models to guide this process. We created a platform for making these semantic enrichments, while allowing them to be reused, extended and nested, which makes us conclude we have reached our goal: a model-driven platform for the enrichment of RDF datasets.

7.3 Contributions

The main contribution of this Master’s Thesis project is the RDF Gears framework itself. It is a framework that allows performing semantic enrichment on any RDF dataset, using a number of auxiliary RDF datasets.

Apart from the framework as a whole, a number of aspects within the framework have been explicitly designed to make the framework more powerful.

Object typing
The object typing mechanism is primarily used to create mappings between resources within the core RDF data and the global schema (auxiliary dataset). Object typing is also used for (1) type checking during the enrichment flow execution, (2) the validation of connections while specifying an enrichment flow and (3) the recommendation of blocks while specifying an enrichment flow.

Reusability by design
Using the RDF Gears framework, all enrichment flows and enrichment steps created by users can be reused. Series of blocks within an enrichment flow can be stored as one compound enrichment step, which opens up possibilities for encapsulating any number of data operations.
Extensibility of framework  The RDF Gears framework allows external systems to be included in the RDF Gears framework. Currently the Silk interlinker tool is included in the RDF Gears framework.

Flexibility in data manipulation paradigms  Thirdly, the RDF Gears framework does allow different data manipulation paradigms to be part of the framework. Different paradigms might include other programming languages but also future Semantic Web standards (like SPARQL 1.1\(^1\)).

7.4 Future work

Following the work presented in this Thesis document, a number of open issues was identified, each of which is now discussed.

Portal and/or market place for enrichment steps  Since RDF Gears is providing the tools to create new content, such as new enrichments steps or even complete flows, it is likely that a community evolves around the RDF Gears framework. If this turns out to be true, enrichment steps and other user-created content might be shared.

In order to make sharing of user content feasible, the importance of attaching URIs to this content should be stressed. A mechanism for controlling uploaded content should be researched.

Federated SPARQL  At the time of writing, Federated SPARQL or Distributed SPARQL is still an open issue, although the DistributedSAIL library is used to provide federated SPARQL capabilities to RDF Gears. Determining the execution order of a federated SPARQL query proves to be hard, since the evaluation of SPARQL queries is NP-complete (and thus intractable) \[^{24, 29}\]. The performance of federated SPARQL engines is still likely to be improved when commonly usable heuristics are found for speeding up federated SPARQL queries. Potential heuristics could use void data\(^2\) describing a dataset to predict a hopefully better performing way of decomposing a federated SPARQL query.

Other possibilities which can be researched regarding query federation include genetic programming \[^{30}\] and the SPARQL-DQP system \[^{1}\].

Enrichment flow optimization  Current implementation of RDF Gears provides a simple approach that allows creating and executing enrichment flows. The execution order that results from the enrichment flow can be reordered, so that more restrictive blocks are executed before less restrictive blocks. One approach that should be re-searched is the prediction of the performance of blocks in the enrichment flow. Combined with blocks of a smaller granularity (i.e. by dividing current blocks into smaller blocks), this could help speeding up the execution of enrichment flow.

The reordering of blocks in the enrichment flow should be performed after the complete enrichment flow has been specified, and before it is being executed.

\[^{1}\text{http://www.w3.org/TR/sparql11-query/}\]
\[^{2}\text{http://vocab.deri.ie/void}\]
Enhanced recommendations  The current implementation of the RDF Gears framework relies on the object typing mechanism to create reliable recommendations. It is possible to determine which blocks are often used together, by analyzing statistics using data mining.

These statistics may help to find patterns such as:

- Enrichment step A and enrichment step B are used together a lot;
- If enrichment step A and B are part of the enrichment flow, enrichment step C is usually also used.

Also other kinds of patterns may be researched in order to improve on the recommendations that the current implementation provides.
Bibliography


Appendices
Appendix A

Used RDF prefixes

This appendix provides an overview of all RDF prefixes that are used in examples throughout this document. Table A.1 shows this overview.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>_:</td>
<td><a href="http://localhost/publications/inproceedings/">http://localhost/publications/inproceedings/</a></td>
</tr>
<tr>
<td>dc:</td>
<td><a href="http://purl.org/dc/elements/1.1/">http://purl.org/dc/elements/1.1/</a></td>
</tr>
<tr>
<td>dcterms:</td>
<td><a href="http://purl.org/dc/terms/">http://purl.org/dc/terms/</a></td>
</tr>
<tr>
<td>dbo:</td>
<td><a href="http://dbpedia.org/ontology/">http://dbpedia.org/ontology/</a></td>
</tr>
<tr>
<td>foaf:</td>
<td><a href="http://xmlns.com/foaf/0.1/">http://xmlns.com/foaf/0.1/</a></td>
</tr>
<tr>
<td>geonames:</td>
<td><a href="http://www.geonames.org/ontology#">http://www.geonames.org/ontology#</a></td>
</tr>
<tr>
<td>rdf:</td>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a></td>
</tr>
<tr>
<td>rdfs:</td>
<td><a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a></td>
</tr>
<tr>
<td>swrc:</td>
<td><a href="http://swrc.ontoware.org/ontology#">http://swrc.ontoware.org/ontology#</a></td>
</tr>
<tr>
<td>vcard:</td>
<td><a href="http://www.w3.org/2006/vcard/ns#">http://www.w3.org/2006/vcard/ns#</a></td>
</tr>
</tbody>
</table>

Table A.1: Overview of used RDF Prefixes

The graphical user interface of the RDF Gears framework also shows prefixed URIs instead of full URIs for convenience. Prefix.cc\(^1\) is an interesting web service developed by DERI, which can transform a prefixed URI into a full URI (lookup) or a full URI into a prefixed URI (reverse lookup\(^2\)). RDF Gears uses this web service to find prefixes for commonly known vocabularies, including the prefixes discussed in Table A.1\(^3\).

\(^1\)http://prefix.cc
\(^2\)http://prefix.cc/reverse
\(^3\)Apart from the base prefix `_:`
Appendix B

Third-party libraries

This appendix provides an overview of external Java libraries used to create the RDF Gears platform. The libraries have been split up into 2 categories, the first being 'user interface libraries' and the second being libraries to enable use of Semantic Web technologies.

B.1 User interface libraries

**SWT and JFace (Eclipse RCP)** The user interface is developed using the SWT\(^1\) library. This library allows creating multi-platform user interfaces. It provides many widgets, and displays the user interface using a native look-and-feel, which helps the familiarization process. JFace\(^2\) has been used to provide message dialogs and wizards and can also be used for a variety of other uses, such as providing content completion in SWT user interfaces. Both SWT and JFace are part of the Eclipse IDE and the Rich Client Platform (RCP\(^3\)).

**Draw2D** The Draw2D\(^4\) library is used to create the canvas on which enrichment flows are constructed in RDF Gears. The Draw2D library allows drawing figures on a canvas and provides functionality such as figures having borders, customly routed connections and composition of images using different layers. Draw2D was used inside RDF Gears to display the blocks and be able to show connections between these blocks. A useful tutorial to start using Draw2D is available at Eclipse.org\(^5\).

---

B.2 SPARQL and RDF libraries

Sesame (DistributedSail)  The Sesame\textsuperscript{6} API library has been used to connect RDF Gears with the Sesame Server, which in the current implementation acts as the dedicated RDF store for RDF Gears. RDF Gears stores imported RDF datasets inside the Sesame Server, as well as the enriched RDF dataset (when an enrichment flow has been executed).

DistributedSAIL\textsuperscript{7} is a project initiated by the University of Koblenz and is a library that supports federated SPARQL queries (i.e. it is able to send queries to other SPARQL endpoints and combine the results). The DistributedSail approach is one of the data retrieval methods used by RDF Gears. It depends on the Sesame Server being used as the dedicated RDF store of RDF Gears. If another RDF store is used instead (like Jena), DistributedSAIL does not function anymore.

Jena (ARQ)  The libraries Jena\textsuperscript{8} and ARQ\textsuperscript{9} together are used in RDF Gears in order to create an alternative way of retrieving remote RDF data. Jena is a framework for creating Semantic Web applications. It provides (separately downloadable) RDF storage libraries, but these are not used by RDF Gears. Jena is able to work with RDF data as well as RDFS and OWL. ARQ is designed to cooperate with Jena, and allows performing SPARQL queries.

Unlike DistributedSAIL, ARQ does not provide federated SPARQL queries, although the spin-off project DARQ\textsuperscript{10} does.

\textsuperscript{6}http://www.openrdf.org/doc/sesame2/users/
\textsuperscript{7}http://www.uni-koblenz-landau.de/koblenz/fb4/AGStaab/Research/systeme/DistributedSPARQL
\textsuperscript{8}http://openjena.org/index.html
\textsuperscript{9}http://openjena.org/ARQ/documentation.html
\textsuperscript{10}http://darq.sourceforge.net/
Appendix C

Create enrichments with RDF Gears

This appendix describes how to create enriched RDF datasets with the RDF Gears framework. First, we will outline how RDF Gears can be found and installed. Secondly, the example dataset will be introduced, which is used to illustrate the RDF enrichment process. Thirdly, the enrichment of the example RDF dataset will be discussed, in a step by step fashion.

C.1 Installing RDF Gears

The first step to installing RDF Gears is to check-out the sources of RDF Gears (sources/folder) from the WIS SVN repository. In the SVN repository a User Manual and Installation Guide are also available for download, as well as the dataset that is introduced in this chapter.

After checking out RDF Gears, import the complete folder that was checked out as an existing project in the Java IDE. The Ant-file (build.xml) contains targets for building RDF Gears on that Operating System.

One needs to be sure the Sesame server is installed, on a Java Servlet container of choice. Then copy the jar-file DistributedSAIL-2.3.1.jar from the folder distributedsail/ to the folder openrdf-sesame/WEB-INF/lib within the applications folder of your installed Java Servlet container. This action is necessary to create a proper Sesame repository for executing distributed SPARQL queries.

C.2 Introducing the example dataset

In this section we introduce an RDF dataset created by Bas Schoenmakers, which contains a list of publications stored in an RDF/XML file. The dataset reflects

1. https://svn.st.wi.tudelft.nl/wis/rdfgears/version_1.0/, authenticate with user-name: guest, password: wis_guest
2. Slight differences exist because SWT provides a different implementation for each OS
3. DistributedSAIL relies on this repository being created properly
4. Note that Bas Schoenmakers converted relational data into RDF, before creating a Linked Data application with it. This RDF dataset is the intermediate result obtained after converting the relational data into RDF data.
C.3 Enriching the example dataset

In this section we will guide the reader through enriching an RDF dataset, by addressing each of the phases in the 4-phase model identified in Section 4.1. We will use the publications RDF dataset that was just introduced. The enrichment scenario that will be performed now replicates the scenario illustrated in Section 4.3.

C.3.1 Import core RDF data

RDF Gears can access the RDF dataset which was stored in previous sessions, but initially, no RDF datasets are loaded. The user interface would look like Figure C.2.

The first step towards enriching RDF datasets, is importing an RDF dataset. One can import a dataset either by selecting 'Import RDF data from file' in the sidebar of RDF Gears, or by selecting 'Import RDF...' in the File menu and then selecting an RDF file to import. All of the four main flavours\(^5\) of RDF are supported for importing.

---

\(^5\)RDF/XML, N3, Turtle and NTriples
Importing RDF data will take longer if the imported RDF file is larger. Also, the previously imported data needs to be cleared before new RDF data can be imported.

We now import the publications database, which is stored in the file 'pub_xml.rdf' (downloadable from the WIS SVN repository).

C.3.2 Specify enrichment flow

After importing our example dataset, RDF Gears will look similar to Figure C.3. The next action is specifying an enrichment flow6.

---

6Note that we can also load an enrichment flow if we have saved one before
In order to enrich the RDF dataset, an output predicate needs to be added to the output predicates block. Further, a connection must be made from the input predicates block to the output predicates block (usually via one or more other blocks).

Let us first add an output predicate to the enrichment flow. Select the output predicates block, choose the ‘Add output predicate’ button in the opened tab. A wizard is opened that helps specifying the output predicate. It lists a number of well-known vocabularies and their predicates. In the current example, we choose the vcard:geo predicate. After confirming the wizard, the new predicate will be added to the output predicates block.

The next step is connecting all blocks. In order to do so, the connection mode needs to be started (by pressing $\text{Ctrl}+\text{c}$). By clicking one port first, and clicking another port next, a connection will be created between the two ports. RDF Gears will make sure the integrity of the connection is guaranteed. Now connect the blocks like shown in Figure C.4.

Two screencasts are also available showing the basic process of making an enrichment flow\(^7\).

![Figure C.4: Connected 'Geonames City lookup' block in enrichment flow](image)

### C.3.3 Execute specified enrichment flow

At this point in time, we notice that the ‘Execute enrichment’ button on the right bottom of the RDF Gears user interface is now enabled. The flow that we constructed is now ready for execution. The flow takes the values for the predicate swrc:address, searches these values using the Geonames search service, and stores the resulting values into the RDF dataset as values of the predicate vcard:geo.

Now let us execute the enrichment flow which was just specified. First, RDF Gears displays a progress window showing how many of the blocks inside the enrichment

---

\(^7\)http://www.youtube.com/watch?v=vKrU-4oIW_g and http://www.youtube.com/watch?v=mpYbg2MZ1o0
flow have been processed (shown on the left in Figure C.5). The execution time depends on both the size of the imported RDF dataset as well as the nature of the blocks which are part of the specified flow.

Figure C.5: Overview of data enrichment and a summary of the results

After the enrichment flow has been executed, a summary is presented (as shown on the right of Figure C.5). The total number of enriched resources is shown, and the number of new RDF statements.

C.3.4 Access enriched RDF data

After the enrichment flow has been executed, we could start specifying a new enrichment flow and execute it. Instead, we will now discuss the options for accessing the enriched RDF dataset. A first observation of the data should make clear the predicate vcard:geo is now included in the original RDF dataset (shown in Figure C.1).

Export the RDF data

The first of the two options is exporting the enriched RDF data to an RDF file. The enriched RDF dataset can be exported by either selecting ‘Export RDF data to file’ in the sidebar of RDF Gears, or by selecting ‘Export RDF...’ in the File menu. A wizard will be opened which allows specifying the file in which the RDF dataset should be stored.

Querying the RDF data

The second way of accessing the enriched RDF data is by performing SPARQL queries on the data. By selecting ‘Data on SPARQL endpoint’ in the sidebar of the RDF Gears user interface, the SPARQL endpoint provided by Sesame is opened in a web browser.

The results of enrichment process can be verified using a SPARQL query. A part of the results of this query is shown in Figure C.6.

The Sesame Workbench, which contains the SPARQL endpoint provided by Sesame, also provides navigation of the enriched RDF data. One can navigate through the dataset by selecting a particular resource or literal.

---

8 RDF data can be exported to RDF/XML, N3, Turtle or N-Triples format.

9 SELECT ?Subject ?Predicate ?Object WHERE {
  FILTER ( ?Predicate = vcard:geo )
}
C.4 Summary

This chapter first described the procedure for installing RDF Gears. We also described the dependencies for working with the current implementation of the RDF Gears framework.

We then introduced an RDF dataset containing publications of the ST department of the University of Technology, Delft, created by Bas Schoenmakers. The dataset consists of publications which are described using the Dublin Core (1.1 and legacy) and SWRC vocabularies. It served as an example dataset for showcasing the RDF Gears framework and adding another triple to the RDF dataset.

Using the four phases of semantic enrichment as identified in Section 4.1, we explained each of the steps necessary to create enriched RDF datasets.