Predicting query performance and explaining results to assist Linked Data consumption

Rakebul Hasan

UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS

ÉCOLE DOCTORALE STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H È S E

pour l'obtention du grade de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis
Mention : Informatique

Présentée et soutenue par

Rakebul Hasan

Predicting Query Performance and Explaining Results to Assist Linked Data Consumption.

Thèse dirigée par Fabien Gandon
et codirigée par Pierre-Antoine Champin

préparée à l'Inria Sophia Antipolis, Equipe WIMMICS

Soutenue le 4 novembre 2014

Jury :

Président :	Johan Montagnat	-	CNRS (I3S)
Rapporteurs :	Philippe Cudré-Mauroux	-	University of Fribourg
	Pascal Molli	-	Nantes University
Directeur :	Fabien Gandon	-	Inria (WIMMICS)
Codirecteur :	Pierre-Antoine Champin	-	Université Claude Bernard Lyon 1

Acknowledgments

Thanks to all the people who have supported me during my doctoral studies. In particular, thanks to Fabien Gandon for his guidance and Pierre-Antoine Champin for all his inputs in the last three years. I am also grateful to the members of ANR Kolflow project for their feedbacks in the project meetings. Thanks to the *Agence Nationale de la Recherche* for funding my doctoral studies through the Kolflow project (ANR-2010-CORD-021-02). Special thanks to Christine Foggia for all the administrative support. Thanks to the WIMMICS team members for making WIMMICS a fun place to work. Thanks to Amel Othmane, Amine Hallili, Corentin Follenfant, Julien Cojan, Jodi Schneider, Luca Costabello, Maxime Lefrancois, and Papa Fary Diallo for all their support and for helping me out during the difficult times. Thanks to Kemele Endris for his contributions in the evaluating explanations part. Thanks to all the anonymous reviewers of my publications for their comments, which really helped me to shape my thesis. Finally, thanks to my family and friends for their support and encouragements along the way. Special thanks to Katharina Reiter, Kathrin Seibt, and Alexis Morris for always being there as friends and for helping me to cope with the difficult times despite the distance.

Abstract

Our goal is to assist users in understanding SPARQL query performance, query results, and derivations on Linked Data.

To help users in understanding query performance, we provide query performance predictions based on the query execution history. We present a machine learning approach to predict query performances. We do not use statistics about the underlying data for our predictions. This makes our approach suitable for the Linked Data scenario where statistics about the underlying data is often missing such as when the data is controlled by external parties.

To help users in understanding query results, we provide provenance-based query result explanations. We present a non-annotation-based approach to generate why-provenance for SPARQL query results. Our approach does not require any re-engineering of the query processor, the data model, or the query language. We use the existing SPARQL 1.1 constructs to generate provenance by querying the data. This makes our approach suitable for Linked Data. We also present a user study to examine the impact of query result explanations.

Finally to help users in understanding derivations on Linked Data, we introduce the concept of Linked Explanations. We publish explanation metadata as Linked Data. This allows explaining derived data in Linked Data by following the links of the data used in the derivation and the links of their explanation metadata. We present an extension of the W3C PROV ontology to describe explanation metadata. We also present an approach to summarize these explanations to help users filter information in the explanation, and have an understanding of what important information was used in the derivation.

Contents

Chapter 1

Introduction

Contents

1.1 Context

The Web is evolving from a Web of Documents to a Web of Data. Thanks to the W3C Linking Open Data initiative, in the recent years we have seen a sharp growth of publishing Linked Data from community driven efforts, governmental bodies, social networking sites, scientific communities, and corporate bodies [Bonatti 2011]. Data publishers from these different domains publish their data in an interlinked fashion using the RDF data model and provide SPARQL endpoints to enable querying their data, which enables creating a global data space. This presents tremendous potential for integrating disparate data to support a new generation of intelligent applications [Schwarte 2011]. Integrating Linked Data by means of querying may include complex workloads with resource intensive queries. Managing these workloads is vital for effective integration of Linked Data. To this end, understanding query behavior prior to query execution can help users such as knowledge base ad-

ministrators or application developers in workload management tasks such as configuration, organization, inspection, and optimization [Mateen 2014]. Furthermore, in the open environment of the Web where heterogeneous Linked Data is exchanged, integrated, and materialized in distributed repositories behind SPARQL endpoints, understanding query result derivations is essential to make trust judgments, to validate or invalidate results [Theoharis 2011, Wylot 2014]. Query result explanations enable this understanding by providing information such as which source triples contributed to results, how these source triples were combined, and which data sets these source triples came from. In addition, applications can consume Linked Data, some of which can be derived by other applications, and reason on their consumed data to produce results or even produce more Linked Data. In this setting, it is essential to explain not only the reasoning by the applications but also the derivations of the consumed data, to help users to understand how results or new Linked Data were derived. This kind of explanations can become very large when applications consume a large amount of data or the consumed data has a long chain of derivations. In this context, providing explanations with details about all the derivations may overwhelm users with too much information. They may want to have the ability to focus on specific parts of an explanation, filter information from an explanation, or get short explanations with important information.

In the next section, we discuss the issues considering the context we provided so far and identify the research questions.

1.2 Research Questions

The overall research question we address in this thesis is:

RQ. How to assist users in understanding query behavior and results in the context of consuming Linked Data?

We break down this question into several sub-questions. First, we address the problem of understanding query behavior in the context of Linked Data. To enable query behavior understanding, we aim at providing predicted query performance metrics to the users. Users such as knowledge base administrators can use this understanding in use-cases such as effective workload management to meet specific Quality of Service (QoS) requirements. The research question in this context is as follows:

RQ1. How to predict query performance metrics on SPARQL endpoints that provide Linked Data querying services?

Second, we address the problem of providing result explanations to assist users in understanding result derivations. This improved understanding may lead to better trust on the system that produces the result. There are two cases for understanding results in the context of consuming Linked Data: SPARQL query results and results produced by applications.

For SPARQL query results, the main challenge is to provide explanations for queries on SPARQL endpoints which are administrated and controlled by external parties. Hence, re-engineering the underlying data model, the query language, or the query processor to generate explanation related metadata during the query processing is not possible in this scenario. In addition, we investigate the impact of query result explanations in the context of consuming Linked Data. The research questions concerning these issues are as follows:

RQ2. How to provide explanations for SPARQL query results on SPARQL endpoints that provide Linked Data querying services?

RQ3. What are the impacts of query result explanations?

For results produced by applications, the main challenge is to provide explanation facilities considering the distributed and decentralized architecture of the Web. Applications can consume data that are distributed across the Web. The consumed

data in this context can be also some derived data. We investigate how to provide explanation in such a scenario – explaining not only the reasoning by the applications but also the derivations of consumed data. Furthermore, providing detailed explanations may overwhelm users with too much information – specially the non-expert users. In this context, the challenge is to summarize explanations to provide short explanations. Considering these issues, the research questions are as follows:

RQ4. How to provide explanations for results produced by applications that consume Linked Data?

RQ5. How to summarize explanations for results produced by applications that consume Linked Data?

1.3 Contributions

We have five major contributions:

- To address the research question **RQ1**, we present an approach to predict SPARQL query performance without using statistics about the underlying data. We learn query performance from previously executed queries using machine learning techniques. We discuss how to model SPARQL query features as feature vectors for machine learning algorithms such as k-nearest neighbors algorithm (k-NN) and support vector machine (SVM). In our experimental setting, we predict query execution time as a query performance metric with high accuracy.

- To address the research question **RQ2**, we present a non-annotation approach to generate *why-provenance* for SPARQL query results. We show the feasibility of our approach by an experiment to generate *why-provenance* for common Linked Data queries. We generate SPARQL query result explanations from the *why-provenance* of query results.

- To address the research question **RQ3**, we present a user study to evaluate the impact of query result explanations. We conduct the study in a federated query processing setting for Linked Data. Our study shows that query result explanations improve users' user experience – where user experience is defined as understanding and trust.

- To address the research question **RQ4**, we present an approach to explain Linked Data – *i.e.* explaining distributed reasoning in decentralized fashion. We present the *Ratio4TA*[1] vocabulary and introduce the notion of Linked Explanations.

- To address the research question **RQ5**, we present an approach to summarize explanations for Linked Data. We presented five measures to summarize explanations and evaluate different combinations of these measures. The evaluation shows that our approach produces high quality rankings for summarizing explanation statements.

1.4 Thesis Outline

This thesis contains a background and state of the art of the related literature, an approach to SPARQL query performance prediction, an approach to explain SPARQL query results, a user study to evaluate the impact of query result explanations, an approach to explain results produced by Linked Data applications, and an approach to summarize explanations for Linked Data applications. The chapters in the rest of this thesis are organized as follows:

⁕ **Chapter 2** provides a background of the related topics, and the state of the art on user assistance in querying and user assistance in result understanding. We identify the research trends in the areas of user assistance in querying and user assistance in result understanding, and outline the focus of this thesis.

[1] http://ns.inria.fr/ratio4ta/

- **Chapter 3** describes our approach to query performance prediction to assist users in understanding query behavior on SPARQL endpoints that provide Linked Data querying services. We present a machine learning approach to predict SPARQL query performance metrics prior to query execution. We discuss how to model SPARQL query features as feature vectors for machine learning algorithms such as k-nearest neighbors algorithm (k-NN) and support vector machine (SVM). We present an experiment with common Linked Data queries and discuss our results.

- **Chapter 4** describes our approach to explain SPARQL query results. We present a non-annotation approach to generate *why-provenance* for SPARQL query result. We present an experiment with common Linked Data queries to show the feasibility of our algorithm. We present an explanation-aware federated query processor prototype and use our *why-provenance* algorithm to generate explanations for its query results.

- **Chapter 5** describes our user study to evaluate the impact of query result explanations in the Linked Data federated query processing scenario.

- **Chapter 6** describes our approach to explain results produced by applications that consume Linked Data. We introduce an ontology to describe explanation metadata and introduce the notion of Linked Explanations – publishing explanation metadata as Linked Data.

- **Chapter 7** describes our approach to summarize explanations produced by applications that consume Linked Data. We discuss our summarization measures and present an evaluation of our summarization approach.

- **Chapter 8** summarizes our contributions and describes our perspectives.

1.5 Publications

The publications resulted from this thesis are as follows:

Query Performance Prediction

1. Rakebul Hasan and Fabien Gandon. A Machine Learning Approach to SPARQL Query Performance Prediction. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence 2014 (WI 2014), August 2014.

2. Rakebul Hasan and Fabien Gandon. Predicting SPARQL Query Performance. Poster, the 11th Extended Semantic Web Conference (ESWC 2014). May 2014.

SPARQL Query Result Explanation

3. Rakebul Hasan, Kemele M. Endris and Fabien Gandon. SPARQL Query Result Explanation for Linked Data. In Semantic Web Collaborative Spaces Workshop 2014 (SWCS 2014), co-located with the 13th International Semantic Web Conference (ISWC 2014), October 2014.

Explanation for Linked Data

4. Rakebul Hasan. Generating and Summarizing Explanations for Linked Data. In Proceedings of the Extended Semantic Web Conference 2014 (ESWC 2014), May 2014.

5. Rakebul Hasan and Fabien Gandon. A Brief Review of Explanation in the Semantic Web. In Workshop on Explanation-aware Computing (ExaCt 2012), co-located with the European Conference on Artificial Intelligence (ECAI 2012), August 2012.

6. Rakebul Hasan and Fabien Gandon. Linking Justifications in the Collaborative Semantic Web Applications. In the Semantic Web Collaborative Spaces

Workshop 2012 (SWCS 2012), co-located with the 21st International World Wide Web Conference 2012 (WWW 2012), April 2012.

Doctoral Symposium

7. Rakebul Hasan. Predicting SPARQL Query Performance and Explaining Linked Data. In PhD Symposium of the Extended Semantic Web Conference 2014 (ESWC 2014), May 2014.

Chapter 2

Background and State of the Art

Contents

2.1 Introduction

In this chapter, we review the topics required for the background knowledge for this thesis and provide a state of the art review of the related literature. We first provide a brief history of the evolution of the Web. Second, we discuss the Linked Data principles with a focus on publishing and consuming Linked Data. Third, we review the literature on user assistance in querying. Furthermore, we review the literature on user assistance in understanding results. Finally, we discuss the research trends and challenges, and the focus of this thesis.

2.1.1 Publication

We published the result of this chapter as a full research (survey) paper [Hasan 2012b] in the Explanation-aware Computing Workshop 2012 (ExaCt 2012) at European Conference on Artificial Intelligence 2012 (ECAI 2012).

2.2 From the Web of Document to the Web of Data

The Web has evolved from its early days of the Web of Documents to the modern Web of Data. Tim Berners-Lee in his original proposal of the "World Wide Web" (WWW) [Berners-Lee 1990] introduced WWW as a hypertext application to cross-link documents all over the world using the Internet. The basic idea of the WWW is that a client application called the Web browser can access a document in another computer by sending a message over the Internet to a Web server application. In response to a client's access request message, the Web server sends back a representation of the document – written in the Hypertext Markup Language (HTML). HTML allows adding hyperlinks to other documents at different locations on the Web. The location of a Web document (Web page) is named using a Universal Resource Locator (URL). When a user clicks on a hyperlink, the Web browser sends a message to the Web server at the IP address associated with the URL, requesting

a representation of the HTML document at the given location from the Web server. The Web server sends back the HTML source code of the requested document and the browser displays it to the user. A turning point for the WWW was the introduction of the Mosaic web browser[1] in 1993. It could display both textual and graphical contents. This lead to a rapid growth of the usage of the WWW. In the core of the notion of the WWW is the idea of an open community: anyone can say anything about any topic (known as the AAA slogan). This openness led to the wider adaption and development of the Web with a comprehensive coverage of topics. However, during the early phases of the WWW, most Web documents were static – with no option for the users to contribute to the content of the documents.

As Simperl *et al* [Simperl 2013] describe, the second phase of WWW development began around 2000 with the introduction of technologies for allowing users to interact with the Web pages and contribute to their contents. This lead to the development and adaptation of a wide range of social websites including blogs, wikis, product reviews, and crowdsourcing. The Web users, previously consumers of the Web contents, became prosumers capable of contributing to the contents of the Web. With this, the AAA slogan became even more prevalent.

In 2001, Berners-Lee *et al.* [Berners-Lee 2001] proposed a further development of the Web called the Semantic Web. They pointed out that the existing Web was not usable by computer applications the same way they are usable by people. For example, a person can look at different Web pages providing textual information on flight schedules, hotels, weather, and so on, and plan a trip. However, reliably extracting such information from text-based Web pages is hard for computer applications. The main aim of the Semantic Web is to support a distributed Web of data rather than a distributed Web of documents. This means that instead of having one Web document link to another Web document, one data item can link to another data item using different types of relations. This enables content providers to publish human-readable Web documents along with machine-readable description

[1] http://www.livinginternet.com/w/wi_mosaic.htm

of the data. With this vision, the Semantic Web initiative resulted in standards for publishing data on the Web and consuming those data to allow computer applications to combine data from different sources the same way a person can combine information from different textual Web pages to perform a task.

In 2006 Berners-Lee proposed a set principles [Berners-Lee 2006a] – known as the Linked Data principles – for publishing data on the Semantic Web. This resulted in a sharp growth of published data on the Semantic Web following the Linked Data principles – from 2 billion triples in 2007 to over 30 billion triples in 2011.

2.3 Linked Data

The term Linked Data refers to a set of best practices – proposed by Berners-Lee in his Web architecture note Linked Data [Berners-Lee 2006a] – for publishing and interlinking data on the Web [Heath 2011]. The basic idea of Linked Data is to use the Web architecture to share Semantic Web data. Before discussing the Linked Data principles, we briefly introduce the RDF data model for representing data on the Semantic Web and the SPARQL query language to query data on the Semantic Web. For a more detailed introduction to RDF and SPARQL, we refer the readers to the cited W3C specification documents [RDF 2014a, SPA 2013b].

2.3.1 RDF

The Resource Description Framework (RDF) data model is a W3C recommended standard for representing information about resources on the World Wide Web [RDF 2014a]. RDF is a graph-based data model where vertices represent entities and edges represent relationships between entities.

Definition 1 (*RDF graph*) *Let I be the set of IRIs, L be the set of literals, and B be the set of blank nodes. An RDF triple (s,p,o) is a member of the set $(I \cup B) \times I \times (I \cup L \cup B)$. An RDF graph is a set of RDF triples. For an RDF triple (s,p,o),*

the element s is called subject, the element p is called predicate, and the element o is called object.

2.3.2 SPARQL

SPARQL is the W3C recommended query language for RDF. As the SPARQL 1.1 specification describes [SPA 2013b], SPARQL query solving is based on graph pattern matching. SPARQL queries allow specifying sets of *triple patterns* known as basic graph patterns. *Triple patterns* are similar to RDF triples but the subject, predicate, and object can be variables. A basic graph pattern may match a subgraph from the RDF data and substitute the variables by RDF terms from the matched subgraph. The native SPARQL query engines perform a series of steps to execute a query [SPA 2013b]. First, parsing the query string into an abstract syntax form. Next, transforming the abstract syntax to *SPARQL abstract query*. Finally, optimizing and evaluating the *SPARQL abstract query* on an RDF data set.

Definition 2 (*SPARQL abstract query*) *A SPARQL abstract query is a tuple (E, DS, QF) where E is a SPARQL algebra expression, DS is an RDF data set, QF is a query form.*

The algebra expression E is evaluated against *RDF graphs* in the RDF data set DS. The query form QF ($SELECT$, $CONSTRUCT$, ASK, $DESCRIBE$) uses the solutions from pattern matching to provide result sets or *RDF graphs*. The algebra expression E includes graph patterns and operators such as *FILTER*, *JOIN*, and *ORDER BY* [2]. SPARQL allows forming graph patterns by combining smaller patterns: basic graph patterns, group graph patterns, optional graph patterns, alternative graph patterns, and patterns on named graphs. A basic graph pattern contains a set of *triple patterns*.

Definition 3 (*Triple pattern*) *A triple pattern is a member of the set:* $(T \cup V) \times (I \cup V) \times (T \cup V)$. *The set of RDF terms T is the set* $I \cup L \cup B$. *The set V is the*

[2] Algebra operators: http://www.w3.org/TR/sparql11-query/#sparqlAlgebra

set of query variables where V is infinite and disjoint from T.

A group graph pattern combines all other types of graph patterns. An optional graph pattern contains a graph pattern which is optional to match for a query solution. Alternative graph patterns provide a means to take union of the solutions of two or more graph patterns. Patterns on named graphs provide a means to match patterns against selected graphs when querying a collection of graphs. The outer-most graph pattern in a SPARQL query is known as the query pattern. A query pattern is a group graph pattern.

2.3.3 The Linked Data Principles

The Linked Data principles were proposed by Berners-Lee in his Web architecture note Linked Data [Berners-Lee 2006a]. The principles are the following:

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 they can discover more things.

The first principle advocates using URIs to identify real world objects (e.g. people, places, and cars) and abstract concepts (e.g. relationships between objects, the set of all red cars, and the color red). The second Linked Data principle advocates combining the use of HTTP – the universal access mechanism of the Web – and URIs to enable dereferencing the URIs of objects and abstract concepts over the HTTP protocol to retrieve descriptions of the objects and abstract concepts. The third Linked Data principle advocates the use of a single data model (RDF) for publishing data to enable different applications to process the data. In addition, data providers may provide access to their data via SPARQL endpoints. This enables

providing search APIs over their data sets. The fourth principle advocates linking any type of things using their URIs. For example, a link may be created between a person and a place. This is analogous to hyperlinks in the Web of documents. However, the links are typed relationships in Linked Data. This enables creating a global data space as the URIs may refer to descriptions of things hosted in different Web servers distributed across the Web.

Indeed, many data publishers have adopted these principles to publish their data on the Web. An important development in this context is the W3C Linking Open Data (LOD) initiative[3] which promotes publishing open data sets as Linked Data – known as the LOD cloud. Figure 2.1 shows the LOD cloud diagram[4]. It shows the data sets that have been published as Linked Data by the contributors of the Linking Open Data project and other individuals and organizations, as of September 2011. A node in this diagram represents a distinct data set. An arc from a data set to another data set indicates that there are RDF links from the data set to the other data set. A bidirectional arc between two data sets indicates that there are outward links between both data sets. Larger nodes correspond to a greater number of triples. Heavier arcs represent a greater number of links between two data sets. As of September 2011, the LOD cloud contains 295 data sets classified into 7 domains totaling 31,634,213,770 triples altogether[5].

2.3.4 Publishing Linked Data

Publishing Linked Data requires adopting the Linked Data principles we discussed in Section 2.3.3. Heath and Bizer [Heath 2011] discuss the design considerations for preparing data to publish them as Linked Data and serving Linked Data for consumers. We outline them in this section.

[3] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

[4] Attribution: "Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/"

[5] http://lod-cloud.net/state/

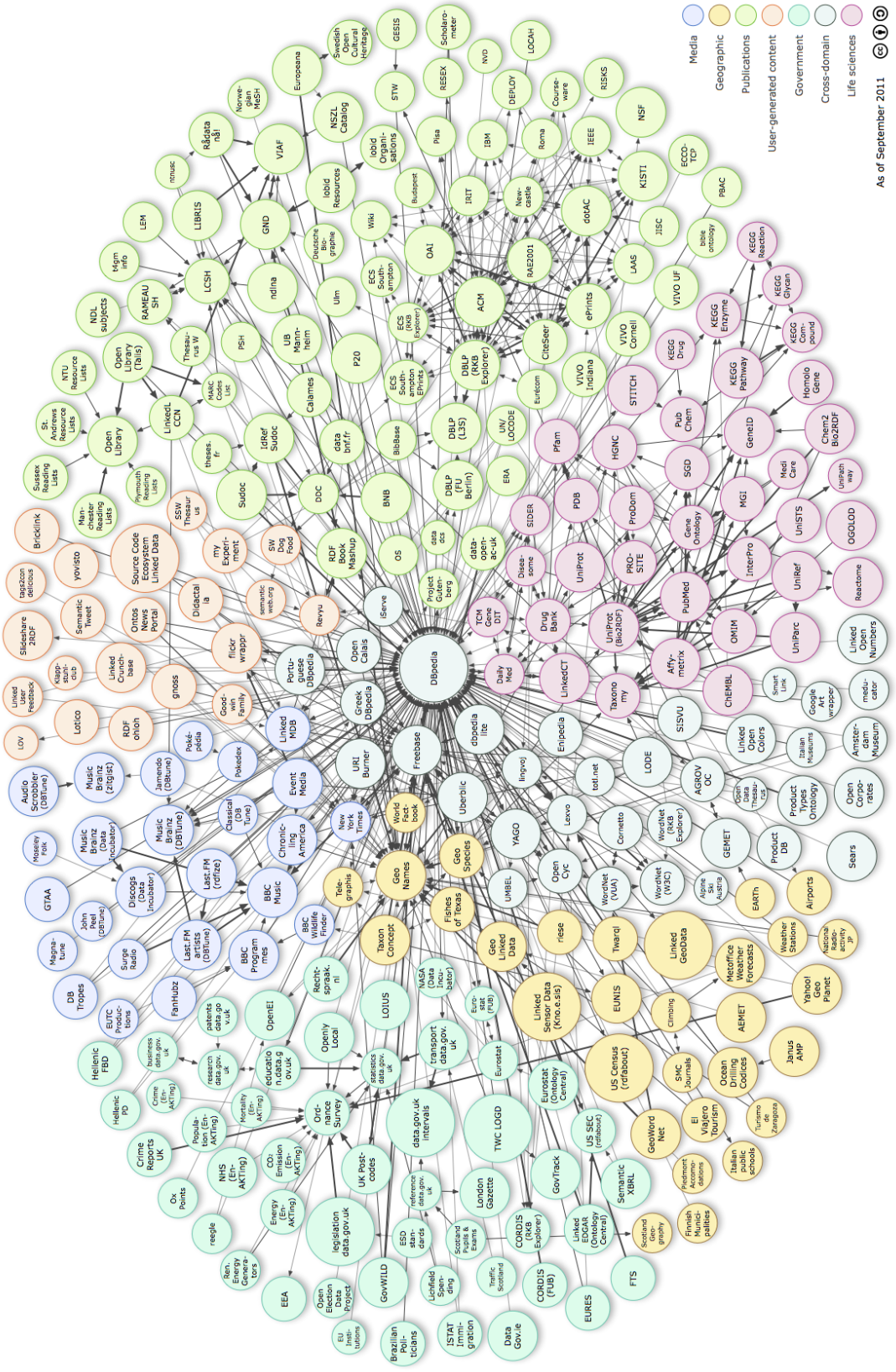


Figure 2.1: Linking Open Data cloud diagram.

2.3.4.1 Design Considerations

Heath and Bizer break down the design considerations for preparing data to publish as Linked Data into three areas: naming things with URIs; describing things with RDF; and making links to other data sets.

Naming Things with URIs. The first Linked Data principle advocates using URIs as names for things. These things can be real-world objects such as a person, a place, a building, or more abstract concepts such as a scientific concept. Names for these things make it possible to refer to each of them. The second Linked Data principle advocates using HTTP URIs to enable names to be looked up by any HTTP client. Using HTTP URIs as names means that a data publisher chooses part of an *http://* namesapce that he/she controls – possibly by owning the domain name, running a Web server for the domain name, and minting URIs in this namespace for naming things. To promote linking to a data set, data publishers follow some simple rules for minting stable and persistent URIs. First, a data publisher should not use a namespace on which he/she does not have control – to enable URI dereferencing. Second, URIs should not include implementation details that may change over time. Finally, creating URIs based on keys that are meaningful in the domain of a data set – e.g. using the ISBN as part of the URI for a book rather than using its internal database key.

Describing Things with RDF. The third Linked Data principle advocates providing useful information when someone looks up a URI. RDF provides an abstract data model for describing resources using triples in a data set. RDF does not provide domain specific terms for describing real world objects and their relationships. For this, taxonomies, vocabularies, and ontologies are used. These taxonomies, vocabularies, and ontologies are expressed in SKOS (Simple Knowledge Organization System) [SKO 2009], RDFS (RDF

Schema) [RDF 2014c], and OWL (Web Ontology Language) [OWL 2014]. SKOS allow expressing conceptual hierarchies, known as taxonomies. RDFS and OWL allows describing conceptual models using classes and properties. Furthermore, it is desirable to reuse terms from existing vocabularies. This makes it easier for applications – which are tuned to well known vocabularies – to consume data. When someone dereference the URI for a resource, the related triples for that resource are provided in the response.

Making Links to Other Data Sets. It is essential to create links within and between data sets to ensure every resource in a data set is discoverable, and that it is well integrated with the Web. It is important that external data sets link to the resources in a new data set published as Linked Data. This allows crawlers and applications to discover newly published data sets. However, third parties owning the external data sets may need convincing about the value of linking to a new data set. DBpedia[6] can be considered as an example of this which allows third parties to include triples with links to their data sets. It is equally important that a new data set links to resources in external data sets. This enables discovering additional data about resources in external data sets by dereferencing their URIs. In addition, those external data sets may include links to some resources in other external data sets, which leads to discovering even more data.

2.3.4.2 Serving Linked Data

The primary mechanism to serve Linked Data is by making URIs defererenceable. In addition, a large number of Linked Data publishers provide SPARQL endpoints for directly querying the data.

Making URIs Defererenceable. HTTP URIs are naturally dereferenceable. HTTP clients can look up a HTTP URI and retrieve the description of the

[6] http://dbpedia.org/

resource that the URI identifies. This mechanism applies to HTTP URIs that identify classical HTML documents, as well as HTTP URIs that identify real-world objects and abstract concepts in the Linked Data context. Resource descriptions are embodied in the form of Web documents. The common practice is to represent the descriptions for human consumption as HTML and the descriptions for machine consumption as RDF data. In fact, data publishers use different URIs to identify a real-word object and the document that describes it, to eliminate ambiguity. This allows making separate statements about an object and about the document that describes it. Different representations of resources are achieved using HTTP content negotiation [Fielding 1999]. The basic idea of content negotiation is that HTTP clients indicate the types of documents they prefer in HTTP headers of each request. Servers select the appropriate representation for the response of a request by inspecting the HTTP header of the request.

SPARQL Endpoints. A SPARQL endpoint is a SPARQL query service via HTTP that implements the SPARQL Protocol [SPA 2013a]. The SPARQL Protocol defines how to send SPARQL queries and update operations to a SPARQL service via HTTP. It also specifies the HTTP responses for a SPARQL query and an update operation. Public SPARQL endpoints serving Linked Data usually do not support the SPARQL update operation. A large fragment of Linked Data is served using SPARQL endpoints. As of September 2011, 68.14% of the data sets (201 out of 295 data sets) in the LOD cloud[7] provide SPARQL endpoints.

2.3.5 Consuming Linked Data

In this section, we outline the aspects related to consuming Linked Data discussed by Heath and Bizer [Heath 2011]. Data published as Linked Data becomes part of

[7] http://lod-cloud.net/state/#access

a global data space. In general, applications use Linked Data from this global data space exploiting the following properties:

Standardized Data Representation and Access. Linked Data is published in a self-descriptive manner, using a standardized data model and standardized data access mechanisms. In comparison to Web 2.0 APIs, data integration becomes easier for Linked Data.

Openness of the Web of Data. The inherently open architecture of Linked Data enables new data source discovery at runtime – automatically discovering new data sources as they become available.

2.3.5.1 Linked Data Applications

Heath and Bizer classifies the current generation of Linked Data applications into two categories: generic applications and domain-specific applications.

Generic Linked Data applications. Generic Linked Data applications process data from any domain. Examples of generic Linked Data applications are: Linked Data browsers and Linked Data search engines. Traditional Web browsers allow users to navigate between HTML Web pages by following hyperlinks. Similarly, Linked Data browsers allow users to navigate between data sources by following links of RDF resources. In this way, a user can begin navigation in one data source and may progressively traverse the Web of Data by following links of RDF resources. Examples of Linked Data browsers include Disco[8], Tabulator[9] [Berners-Lee 2006b], and LinkSailor[10]. Linked Data search engines crawl Linked Data from the Web, and provide query interfaces over the aggregated data. Examples of Linked Data search engines include Sig.ma[11] [Tummarello 2010], Falcons[12] [Cheng 2009], and Semantic Web

[8] http://wifo5-03.informatik.uni-mannheim.de/bizer/ng4j/disco/
[9] http://mes.github.io/marbles/
[10] http://linksailor.com/nav
[11] http://sig.ma/
[12] http://iws.seu.edu.cn/services/falcons/documentsearch/

Search Engine (SWSE)[13] [Harth 2008]. The aim of these services is to provide crawling and indexing infrastructure for Linked Data applications – so that each application does not have to implement them. Services with slightly different emphases include Sindice[14] [Tummarello 2007] which provides access to documents containing instance data; and Swoogle[15] and Watson[16] which provide query interfaces to find ontologies.

Domain-specific applications. There are various Linked Data applications covering specific user communities. The websites data.gov[17] and data.gov.uk[18] provide lists of Linked Data applications which combine and visualize government data to increase government transparency. dayta.me[19] and paggr[20] are examples of Linked Data applications for personal information management and recommendation. Talis Aspire[21] [Clarke 2009] is an example of Linked Data application for education domain which helps users to create and manage learning materials. Other examples of domain-specific Linked Data applications include DBpedia Mobile[22] [Becker 2009] for tourism domain; NCBO Resource Index[23] and Diseasome Map[24] for Life Science domain; and Researcher Map[25] for social networks domain.

2.3.5.2 Architecture of Linked Data Applications

Heath and Bizer discuss three architectural patterns for Linked Data applications: the crawling pattern, the on-the-fly dereferencing pattern, and the query federation

[13] http://www.swse.org/
[14] http://sindice.com/
[15] http://swoogle.umbc.edu/
[16] http://kmi-web05.open.ac.uk/Overview.html
[17] http://www.data.gov/communities/node/116/apps
[18] http://data.gov.uk/apps
[19] http://dayta.me/
[20] http://paggr.com/
[21] http://www.w3.org/2001/sw/sweo/public/UseCases/Talis/
[22] http://wiki.dbpedia.org/DBpediaMobile
[23] http://bioportal.bioontology.org/resources
[24] http://diseasome.eu/map.html
[25] http://researchersmap.informatik.hu-berlin.de/

pattern.

The Crawling Pattern. This pattern mimics the crawling pattern of classical Web search engines. Applications first crawl the Web of Data by traversing links of RDF resources, then they integrate and cleanse the crawled data, and provide and integrated view of the crawled data. The advantages of the crawling pattern is twofold: new data is discovered at run-time and complex queries over the large amount of integrated data can be executed with a reasonable performance. The disadvantage of the crawling pattern is that applications need to replicate the data locally and they often work with stale data.

The On-The-Fly Dereferencing Pattern. A typical use-case for this pattern is implementing a Linked Data browser application. The applications that implement this pattern dereference URIs and follows RDF resource links the moment they require the data. The advantage of this pattern is that applications always process fresh data. The disadvantage of this pattern is that complex operations might require dereferencing a large number of URIs and hence they are slow.

The Query Federation Pattern. The applications that implement this pattern directly send queries (or parts of queries) to a fixed set of SPARQL endpoints – therefore this pattern can be only implemented if the data sources provide SPARQL endpoints in addition to dereferenceable URIs. The advantage of this pattern is that applications do not need to replicate the data locally and hence they always process fresh data. A major problem in this pattern is that finding efficient query execution plans over large number of SPARQL endpoints is difficult – causing significant downgrade in performance when the number of SPARQL endpoints grows. Therefore, this pattern is suitable for scenarios where the number of data sources – SPARQL endpoints – is small.

2.4 User Assistance in Querying

Assisting users in querying has been studied from different point of views. Stojanovic *et al.* [Stojanovic 2004] propose a query refinement approach to help users refine queries according to their needs in a step-by-step fashion. The authors argue that this approach is suitable for modeling information retrieval tasks on ontology based systems. Nandi *et al.* [Nandi 2007] present an automatic query completion approach for relational and XML databases to help users construct queries without prior knowledge of the underlying schema. This approach helps the users to construct queries, while they type, by suggesting schema level parameters and text fragments from the data. Zenz *et al.* [Zenz 2009] introduce the QUICK system to help users construct semantic queries from keywords. It enables a user to start with arbitrary keywords and incrementally constructs the intended query. These approaches help users to formulate and refine queries.

Another line of work on assisting users in querying focuses on helping users in understanding query behaviors prior to query execution. Generally speaking, these works provide query performance predictions based on the query execution history. In the database literature, Duggan *et al.* [Duggan 2011], Akdere *et al.* [Akdere 2012], Ganapathi *et al.* [Ganapathi 2009], and Gupta *et al.* [Gupta 2008] study query performance prediction to support database users in tasks such as Quality of Service (QoS) management and effective resource allocation. For example, database administrators can use query performance prediction to effectively allocate workloads such that specific QoS targets are met. System architects can use query performance prediction to estimate system configurations for supporting some specific kind of workload requirements. Application programmers can use query performance prediction to choose among alternative queries based on performance requirements. These approaches in the database literature study how to accurately predict performance metrics for relational database queries – in human understandable units (e.g. time units for latency) in contrast to some abstract numbers in query cost estimation ap-

proaches for query optimization. Such approaches for query performance prediction to support users have not been studied for Semantic Web queries.

2.5 User Assistance in Understanding Results

Expert systems were among the first software systems that provided features – explanation facilities – for assisting users in understanding how and why the systems produce their results or reach a conclusion [Haynes 2001, Moore 1988, Swartout 1991]. Explanation facilities in expert systems have evolved from reasoning trace oriented explanations, primarily useful for developers and knowledge engineers, to more user oriented interactive explanations justifying why a system behavior is correct, to casual explanations generated in a decoupled way from the line of reasoning. Explanation facilities in expert systems were motivated by enabling transparency in problem solving, imparting an understanding of why and how a given conclusion was reached, and hence enabling trust in the reasoning capabilities of expert systems. These developments motivated adaptation and development of explanation facilities in other fields such as machine learning [Glass 2011, Stumpf 2007], case-based reasoning [Doyle 2003, Roth-Berghofer 2004], recommender systems [Tintarev 2007], databases [Cheney 2009], and Semantic Web. Here we first briefly discuss the general explanation approaches in the Semantic Web context. Then we briefly discuss explanation for query results.

2.5.1 Explanation in the Semantic Web

Generally speaking, the main goal of providing explanations for Semantic Web applications is to improve users' understanding of the process of deriving new information and the flow of information involved in the process. This improved understanding may lead to better user acceptance, and hence improved trust in the Semantic Web applications. The previous work on explanations in the Semantic Web literature can be categorized into two categories: (a) representing explanation metadata, (b)

generating and presenting explanations.

2.5.1.1 Representing Explanation Metadata

A large body of previous work [McGuinness 2004, McGuinness 2006, Pinheiro da Silva 2006, Pinheiro da Silva 2008, Kagal 2011, Bizer 2007, Forcher 2010] has used Semantic Web standards to represent machine processable explanation metadata. Typically explanation metadata include details on information manipulation steps and their dependencies. McGuinness *et al.* termed these kind of metadata as justifications: a justification can be a logical reasoning step, or any kind of computation process, or a factual assertion or assumption [McGuinness 2006, McGuinness 2008, McGuinness 2004] . An important previous work for representing explanation metadata is Proof Markup Language (PML) [Pinheiro da Silva 2006][26]. PML is an explanation interlingua consisting of three OWL ontologies: PML provenance ontology (PML-P), PML justification ontology (PML-J), and PML trust ontology (PML-T). PML-P provides primitives for representing real world things (e.g. information, documents, people) and their properties (e.g. name, creation date-time, description, owners and authors). PML-J provides primitives for encoding justifications for derivations of conclusions. PML-T provides primitives for representing trust assertions concerning sources and belief assertions concerning information. There are also variants of PML: PML-Lite[27] and **A**ccountability **I**n **R**DF (AIR) [Kagal 2011]. PML-Lite is a simplified subset of three PML modules to represent provenance of data flows and data manipulations. AIR rule language includes the AIR Justification Ontology (AIRJ) – an extension of PML-Lite – to represent justifications that the AIR reasoner produces. AIRJ extends the PML-Lite event-based approach. *WIQA - Web Information Quality Assessment Framework* [Bizer 2007] provides explanations in natural language for human consumption and explanations in RDF for further

[26] http://inference-web.org/2007/primer/
[27] http://tw.rpi.edu/web/project/TAMI/PML-Lite

processing by software applications. WIQA describes the explanation trees (parts and subparts of an explanation) using the Explanation (EXPL) Vocabulary[28]. The KOIOS [Forcher 2010] keyword-based semantic search engine provides its search results with explanations about how it computes the search results. KOIOS uses three ontologies to describe its explanations in RDF: KOIOS Process Language (KPL), Mathematical Graph Language (MGL), and Graph Visualization Language (VGL). KPL provides primitives to describe the behavior of the problem solving process. MGL provides primitives to describe the graph based view of the process model. VGL provides primitives to describe visualization parameter related information.

2.5.1.2 Generating and Presenting Explanations

We categorize the previous work on generating and presenting explanations into two categories: explanation-aware applications and justifications.

Explanation-Aware Applications. Inference Web [McGuinness 2003, McGuinness 2004, McGuinness 2008] provides an explanation infrastructure which addresses explanation requirements for web services discovery, policy engines, first order logic theorem provers, task execution, and text analytics. It generates the explanation metadata during the reasoning process and encodes them using PML. Inference Web provides a set of software tools and services for building, presenting, maintaining, and manipulating PML proofs. It proposes a centralized registry based solution for publishing explanation metadata from distributed reasoners. OntoNova [Angele 2003] is an ontology-based question answering system which provides explanations in natural language with its answers. It generates explanations in a meta-inferencing step. The OntoNova inference engine produces log files which represent proof trees for answers. These files are given as an input to a second meta-inference step. This second meta-inference step explains the proof trees in natural language with the description of how answers were derived. WIQA [Bizer 2007]

[28] http://www4.wiwiss.fu-berlin.de/bizer/triqlp/

generates its explanation metadata in RDF during the reasoning process along with natural language annotations using explanation templates to provide the final natural language-based explanation. Antoniou *et al.* [Antoniou 2007, Bassiliades 2007] present a nonmonotonic rule system based on defeasible logic which is able to answer queries and provide proof explanations. The traces of the underlying logic engine are transformed to defeasible logic proofs. The authors introduce an extension to RuleML[29], a unifying family of Web rule languages, to enable formal representation – not in RDF however – of explanations of defeasible logic reasoning. In addition, the authors present graphical user interfaces to visualize the proofs and interact with them. The Knowledge in a Wiki (KiWi) [Kotowski 2010] project[30] provides explanations to support users' trust and determine main causes of inconsistencies in the knowledge base. KiWi generates and stores the justifications of all the derivations during the reasoning process, and uses them for providing explanations and reason maintenance. KiWi provides natural language and proof tree-based explanations highlighting the derivation paths. KOIOS [Forcher 2010] explanations justify how search keywords are mapped to concepts in the underlying RDF data and how the concepts are connected. KOIOS generates the explanation metadata in RDF during the query solving process and presents them as graphical and textual explanations. AIR reasoner [Kagal 2011] generates AIR justifications during its reasoning process. It then converts the AIR justifications in RDF to natural language explanations using user specified translation rules. AIR provides features to selectively control the degree of details in its explanations.

Justifications. Ontology editors such as Protégé[31] and SWOOP[32] provide justification-based explanations for entailments in ontologies. Intuitively, a justification for an entailment is "a minimal subset of the ontology that is sufficient for the entailment to hold" [Horridge 2008]. Horridge [Horridge 2011] pro-

[29] http://ruleml.org
[30] http://www.kiwi-project.eu/
[31] http://protege.stanford.edu/
[32] https://code.google.com/p/swoop/

vides an overview of the justification computation approaches for ontologies. Horridge describes the algorithms for computing justifications using two axes: *single-all-axis* and *reasoner-coupling-axis*. The *single-all-axis* concerns the algorithms to compute a *single* justification and *all* justifications for an entailment. Algorithms for computing *all* justifications generally depend on algorithms for computing *single* justifications. *Single* justifications are useful in application scenarios where human users use the explanations for ontology debugging. The *reasoner-coupling-axis* concerns the explanation generation methods: *black-box* and *glass-box*. *Black-box* [Kalyanpur 2007, Horridge 2009, Wang 2005] methods are reasoner independent. They use the reasoner only to check if an entailment holds. *Glass-box* [Kalyanpur 2005, Meyer 2006, Schlobach 2003, Lam 2008] methods compute justifications as a direct consequence of reasoning. *Glass-box* algorithms usually require modifications of the procedures inside the reasoner in order to generate justifications as a direct consequence of reasoning. There are also hybrid methods [Moodley 2010, Kalyanpur 2005] that combine *black-box* and *glass-box* methods. For example, Kalyanpur *et al.* use a preprocessing *glass-box* algorithm which extracts a small subset of the ontology that entails the entailment, in a *black-box* algorithm to generate the actual justification.

2.5.2 Explaining Query Results

Previous work in the relational database literature suggests explaining query results by providing query result provenance [Cheney 2009]. The general idea of query result provenance is to determine what data or transformations led to result tuples [Herschel 2010]. Data provenance for query results has been widely studied in relational database literature. Recent work (e.g. [Theoharis 2011, Wylot 2014]) in the Semantic Web literature has also studied data provenance for SPARQL query results. In this section we provide an overview of related work on provenance for query results in the database literature and in the Semantic Web literature.

2.5.2.1 Provenance for Query Results in Relational Databases

Cheney *et al.* [Cheney 2009] describe the research trends of provenance in relational database literature. Provenance information explains the origins and the history of data. With the emergence of data on the Internet, where there is no centralized control over the integrity of the data, providing provenance information became increasingly important to help users judge whether query results are trustworthy. Common forms of database provenance describe the relationship between the output and the data in the source. Examples of such provenance information are *why*, *how*, and *where* provenance. *Why-provenance* [Buneman 2001, Cui 2000b] explains why an output was produced. *How-provenance* [Green 2007a] explains how an output was produced. *Where-provenance* [Buneman 2001, Wang 1990] explains where the data in input came from.

Why-provenance. For each tuple t in the output of a query, Cui *et al.* [Cui 2000b] associate a set of tuples in the input – called *lineage* of t. Intuitively, the *lineage* of an output tuple t for a query Q is the input data that contribute to producing t. The lineage of an output tuple acts as the *witness* for the existence of the output tuple. However, not every tuple in the lineage is necessary for the output tuple – there can be multiple witnesses in the lineage for an output tuple. Buneman *et al.* [Buneman 2001] formalize this notion by introducing *why-provenance* that captures different witnesses. For a query Q and output tuple t, a *witness* is a sufficient subset of the database records which ensures that the tuple t is in the output. Buneman *et al.* show that the number of witnesses can be exponential in the size of input database and describe *why* provenance as *witness basis* which restricts to a smaller number of witnesses. *Witness basis* of an output tuple t for a query Q on a database D is a particular set of witnesses which can be calculated efficiently and every witness contains an element of the *witness basis*.

How-provenance. *Why-provenance* describes the source tuples that witness the existence of an output tuple for a query. But it does not explain the structure of the proof of the derivation process – e.g. how many times a tuple contributes to the output tuple. Therefore, *why-provenance* does not explain *how* an output tuple is derived for a query. Green *et al.* [Green 2007a] formalize a notion of *how-provenance* by representing the provenance of an output tuple as a polynomial – known as *provenance semirings* – which describes the structure of the proof by which the output tuple is derived. Interestingly, it is possible to derive *why-provenance* of an output tuple from its *how-provenance*. However, the converse is not always possible.

Where-provenance. Buneman *et al.* [Buneman 2001] also introduce *where-provenance* which describes the relationship between source and output *locations* – a location is the column of a tuple in relational databases. The *where-provenance* of a value in a *location* l in the result of a query Q on database D consists of the *locations* of D from which the value in *location* l was copied according to Q. Buneman *et al.* show that the *where-provenance* of a value v of an output tuple t consists of *locations* found in the *why-provenance* of t. An interesting application of *where-provenance* is the study of annotation propagation [Buneman 2001, Wang 1990]. We can view a notion of provenance as a method to propagate annotations from the input to the output. Similarly, we can view a notion of annotation propagation as a form of provenance by annotating each part of the input with distinct annotations and observing where the annotations end up in the output.

Concerning computing provenance in databases, there are two approaches: the *eager* approach (also known as the *bookkeeping* or *annotation* approach) and the *lazy* approach (also known as the *non-annotation* approach). In the *eager* approach, the original query or the transformation process is re-engineered to carry over extra annotations in the output. The provenance information is derived by examining

the extra annotations and the output. The *eager* approach has a performance overhead and a storage overhead as extra work is done for generating and storing additional annotations. The advantage of the *eager* approach is that provenance can be directly derived for the output and the extra annotations, without examining the source database. Notable examples of *eager* approach implementations are the ORCHESTRA [Green 2009, Green 2007b] and Trio [Agrawal 2006] systems for *how-provenance*, and DBNotes [Bhagwat 2005] for *where-provenance*. In the *lazy* approach, provenance is computed when it is needed, by examining the output and the source data. Therefore, the *lazy* approach can be deployed on an existing database system without having to re-engineer the system. The *lazy* approach does not have performance or storage overheads. However, it is not possible to use the *lazy* approach in scenarios where the source data becomes unavailable. Notable examples of the *lazy* approach are WHIPS [Cui 2000a, Cui 2000b] for *why-provenance* and SPIDER [Alexe 2006, Chiticariu 2006] for *how-provenance*.

In addition to the types of provenance mentioned above, recent work has focused on explaining missing answers – also known as *why-not* provenance. These approaches explain why a tuple is not in the result. Huang *et al.* [Huang 2008] provide provenance for potential answers and never answers by examining if tuple insertions or modifications can yield the desired result. Tran *et al.* [Tran 2010] focus on what modification in the query would yield including the missing tuple in the result. Meliou *et al.* [Meliou 2009] present *why-not* provenance based on causality which combines both tuple modification and query modification approaches.

2.5.2.2 Provenance for Query Results in the Semantic Web

Recent W3C standardization activity on provenance has led to the W3C PROV Ontology [Moreau 2013] recommendation for interchanging provenance information, which considers the overlap in the previous work on representing provenance. A large body of work on provenance in the Semantic Web community has focused on

designing models to represent provenance information [Wylot 2014]. Some previous works (e.g. [Buneman 2010, Flouris 2009]) have focused on extracting provenance for RDF(S) entailed triples, but do not support extracting provenance for SPARQL query results. Some recent works have focused on extracting provenance for SPARQL query results. We overview these recent approaches below.

Theoharis *et al.* [Theoharis 2011] investigate how relational provenance approaches can be applied for SPARQL query result explanations. Theoharis *et al.* represent RDF triples as a relational table with (*subject, predicate, object*) columns, then store the triples in a relational database, and finally query them using a subset of relational algebra called positive relational algebra (RA^+) – this subset excludes the relational algebra difference operator. This transformation allows the authors to use provenance models for relational databases that we discuss in section 2.5.2.1. The authors show that there is an analogy of the SPARQL algebra projection, filter, join, and union operators with the corresponding RA^+ operators. The authors define this fragment of SPARQL algebra operators as positive SPARQL ($SPARQL^+$) and support *why-provenance* and *how-provenance* for $SPARQL^+$. The authors also discuss the limitations of the provenance models for relational databases in capturing the semantics of SPARQL OPTIONAL operator, which implicitly introduces a notion of negation.

Similar to the approach of Theoharis *et al.*, Damásio *et al.* [Damásio 2012] adapt the seminal works on provenance for relational databases. The approach of Damásio *et al.* is based on translating SPARQL queries into relational queries and translating the input RDF graph to a ternary relation with annotation to provide *how-provenance* for SPARQL query results. In contrast to the work of Theoharis *et al.*, the authors consider a significant fragment of SPARQL 1.1 operators, including non-monotonic constructs (OPTIONAL, MINUS, and NOT EXISTS). The authors refute the claim of Theoharis *et al.* that the existing provenance models for relational databases cannot capture the semantics of SPARQL OPTIONAL operator.

Wylot *et al.* [Wylot 2014] present an RDF store called TripleProv which can process provenance-enabled SPARQL queries. The authors work with the notion of provenance polynomial. This work presents storage models for compact representation of provenance data in native RDF stores. The authors also discuss query processing strategies to derive provenance polynomials while processing the query. However, it is not clear which fragment of SPARQL query operators this work supports.

Corese/KGRAM[33] [Corby 2012] SPARQL query engine keeps track of the matched triples for basic graph patterns for a query as part of the query solving process. This way Corese/KGRAM provides provenance information for query results. The provenance feature supports a significant fragment of SPARQL 1.1 operators, including OPTIONAL and property paths, and excluding subqueries, minus and exists filters.

Other notable work on provenance for SPARQL include [Dividino 2009, Zimmermann 2012]. Dividino *et al.* [Dividino 2009] present an extension of RDF to represent meta information, focusing on provenance and uncertainty. The authors use named graphs to store the meta information and provide an extension of SPARQL that enables querying the meta information. Zimmermann *et al.* [Zimmermann 2012] present a framework for annotated RDF. The authors discuss how provenance information can be modeled as annotations using their framework. The authors provide an extension of SPARQL to query RDF with annotations. The query language exposes annotations at query level using annotation variables.

2.6 Discussion

In this section, we discuss the research trends and challenges related to the works we have discussed so far.

[33] http://wimmics.inria.fr/corese

2.6.1 User Assistance in Querying

As we discuss in Section 2.3.4, a large amount of Linked Data is accessible by SPARQL endpoints. In this context, the challenge is to understand how to assist users in querying Linked Data. The reviewed work intends to assist users in querying in three aspects: query refinement, query construction, and query behavior understanding.

2.6.1.1 Query Refinement

Stojanovic *et al.* [Stojanovic 2004] present an approach for query refinement in ontology-based systems. The authors focus on conjunctive queries for ontology-based information retrieval systems. The main goal of this work is to support users to navigate through information contents incrementally and interactively. The main challenge in this line of work is to find the refinements for a query. Stojanovic *et al.* consider the query refinement problem as the problem of inferring all the subsumed queries for a given query. Another challenge is to rank the query refinements. Stojanovic *et al.* rank query refinements according to user's needs and behaviors. In this direction, modeling and analyzing user's needs and behaviors is another challenge.

2.6.1.2 Query Construction

Nandi *et al.* [Nandi 2007] present an approach which helps users to incrementally and instantaneously formulate conjunctive attribute-value queries for relational and XML databases. Zenz *et al.* [Zenz 2009] present a similar approach which helps users to construct queries from keywords for ontology bases systems. These approaches allow users to start with an arbitrary key/keyword and guide users to incrementally construct the intended query by providing them suggestions in the involved steps. The main research problem in this context is to infer what users expect while writing a query. In addition, ranking and presenting the suggestions effectively is also crucial.

2.6.1.3 Query Behavior Understanding

Previous work in the database literature [Duggan 2011, Akdere 2012, Ganapathi 2009, Gupta 2008] presents approaches for predicting query performance metrics for relational databases to help users prior to query execution in understanding how queries behave. The aim of these works is to help users in workload management to meet specific QoS requirements by providing predicted query performance metrics. The main challenge in this line of work is to predict query performance before executing the queries. Previous work uses machine learning techniques to learn query performance (e.g. latency). In the context of Linked Data, the challenge is to predict query performance from the querying side – without using any statistics about the underlying data as they are often missing [Tsialiamanis 2012]. An effective solution to this problem is to learn query performance from query logs of already executed queries, which we discuss in Chapter 3.

2.6.2 User Assistance in Understanding Results

As we discuss in Section 2.5, there is a large literature on helping users to understand results by providing explanations. These explanations may include information manipulation steps by algorithms, proof trees of derivations, justifications for entailments, and provenance for query results.

2.6.2.1 Explanation-Aware Semantic Web Applications

Table 2.1 shows a comparison of explanation-aware Semantic Web applications based on the following criteria.

Metadata Representation. Exposing explanation metadata as RDF enables external software applications to process and make sense of the explanations. This is especially important in the Linked Data scenario where data consumers can also consume explanation metadata if they are published as Linked Data.

What is Explained. The reviewed research discusses explaining the reasoning process (information manipulation steps and operations) and explaining derivations of results.

Explanation Content. Reflects what type of contents are included in an explanation. The reviewed research discuss providing explanations with information about reasoning processes and proof trees of derivations.

Generation. Reflects how explanations are generated.

Presentation. Reflects what kind of user interface presentations are provided. The reviewed research discuss natural language based explanations and graphical explanations.

Summarization. Reflects whether the work supports summarizing explanations. Explanations often can be overwhelming. It is important to provide features to filter information in explanations and summarize important information in explanations to deal with the overwhelming scenarios.

Evaluation. Reflects whether the work evaluates the impact of explanation on users.

Table 2.1 shows that not all the reviewed approaches expose explanation metadata using RDF. This is an undesirable situation in the context of Linked Data. Inference Web, WIQA, and KOIOS explain steps in their reasoning processes and why their results were derived. They provide information about the steps of their reasoning processes and show proof trees of derivations. AIR, OntoNova, Antoniou *et al.*, Bassiliades *et al.*, and KiWi only explain why their results were derived by providing proof trees of derivations. All the reviewed works generate explanations from the reasoning traces. This means these applications are engineered to generate traces of their reasoning steps. Inference Web, KOIOS, and KiWi provide both natural language and graphical presentations in their explanations. AIR and

	Inference Web [McGuinness 2008]	AIR [Kagal 2011]	WIQA [Bizer 2007]	KOIOS [Forcher 2010]	OntoNova [Angele 2003]	[Antoniou 2007, Bassiliades 2007]	KiWi [Kotowski 2010]
Metadata Representation	√	√	√	√			
What is Explained	*R, D*	*D*	*R, D*	*R, D*	*D*	*D*	*D*
Explanation Content	*R, P*	*P*	*R, P*	*R, P*	*P*	*P*	*P*
Generation	*T*	*T*	*T*	*T*	*T*	*T*	*T*
Presentation	*NL, G*	*NL*	*NL*	*NL, G*	*G*	*G*	*NL, G*
Summarization	√						
Evaluation	√						

Table 2.1: Comparison of explanation-aware Semantic Web application approaches. *R* denotes reasoning processes, *D* derivations, *P* denotes proof trees, *T* denotes reasoning traces, *NL* denotes natural language, *G* denotes graphical, √ denotes full support, and empty cell denotes no support

	[Horridge 2011]	[Kalyanpur 2007, Horridge 2009, Wang 2005]	[Kalyanpur 2005, Meyer 2006, Schlobach 2003]	[Lam 2008, Moodley 2010]	[Buneman 2010]	[Flouris 2009]
Metadata Representation						
Generation	*B, G*	*B*	*G*	*B, G*	*G*	*G*
Summarization	√					

Table 2.2: Comparison of justification based approaches. *B* denotes black-box, *G* denotes glass-box, √ denotes full support, and empty cell denotes no support

WIQA provide only graphical presentation, whereas OntoNova, Antoniou *et al.*, and Bassiliades *et al.* provide only graphical explanations. Only Inference Web provides a summarization feature in their graphical explanations by means of zooming in for more details in the proof trees and zooming out for less details. Only Inference Web provides evaluations for their explanations for geospacial domain. Inference Web provides a user study to verify whether explanations play a role for scientists to understand uncertainties related to geospatial information.

2.6.2.2 Justifications

Table 2.2 shows a comparison of approaches to generate justifications for entailments using three criteria: metadata representation, generation, and summarization.

None of the reviewed works on justification exposes explanation metadata using RDF. There are two approaches to generate justifications: black-box and glass-box.

	[Theoharis 2011]	[Damásio 2012]	[Wylot 2014]	[Corby 2012]	[Dividino 2009]	[Zimmermann 2012]
Metadata Representation						
Types of Provenance	*W, H*	*H*	*H*	*W*	*H*	*H*
Generation		*A*	*A*	*A*	*A*	*A*

Table 2.3: Comparison of approaches for SPARQL query result provenance. *W* denotes *why-provenance*, *H* denotes *how-provenance*, *A* denotes annotation approach, and empty cell denotes no support

Black-box approaches are independent of the underlying reasoner. Justifications are computed when they are needed. Black-box approaches can be deployed without having to re-engineer the underlying system. Glass-box approaches require re-engineering the underlying system. Glass-box approaches are harder to implement, but the justifications are computed as a direct consequence of reasoning. Horridge presents laconic and precise justifications which are fine-grained justifications consisting of axioms with no superfluous part. These fine-grained justifications can be seen as summarized justifications. The authors present an optimized algorithm to compute laconic justifications showing the feasibility of computing laconic justifications and precise justifications in practice.

2.6.2.3 Query Result Provenance

Table shows a comparison of approaches for SPARQL query result provenance using three criteria: metadata representation, types of provenance, and generation. The "types of provenance" criterion reflects what type of provenance is supported.

None of the reviewed works on justification exposes explanation metadata using RDF. Only Theoharis *et al.* and Corby *et al.* support *why-provenance*. All the works except the work of Corby *et al.* support *how-provenance*. It is noticeable that SPARQL provenance related works do not define *where-provenance*. This is due to

the difference between the rational data model and RDF data model. The notion of columns does not exist in RDF, which is the key concept of *where-provenance*. As we discuss in Section 2.5.2.1, there are two approaches to generate justifications: annotation approach (also known as eager approach), and non-annotation approach (also known as lazy approach). Theoharis *et al.* do not discuss generation of provenance. Their work is on the theoretical aspects of SPARQL query result provenance. All the other reviewed works support annotation approaches for generating provenance.

2.6.3 The Focus of this Thesis

The focus of this thesis is twofold:

i. Assisting users in understanding query behavior on Linked Data prior to query execution.

ii. Assisting users in understanding query results on Linked Data and results produced by Linked Data applications.

Concerning query behavior understanding, the goal is to assist users in tasks such as workload management to meet specific QoS requirements by providing predicted query performance metrics. The main challenge in this regard is to predict query performance metrics prior to query execution for queries on SPARQL endpoints. Traditional SPARQL query cost estimation techniques such as [Stocker 2008] are based on statistics about the underlying data. However, statistics about the underlying data are often missing in Linked Data [Tsialiamanis 2012]. As of September 2011, only 32.2% of the data sets in the LOD cloud provide basic statistics about their underlying RDF data[34]. In addition, these statistics are often not detailed enough for query cost estimation models. We investigate how to predict SPARQL query performance metrics for queries on SPARQL endpoints without using underlying data statistics.

[34]http://lod-cloud.net/state/

Concerning result understanding, the goal is to provide users information about the process of result derivation to enable them make better trust judgments. We investigate how to provide explanations for SPARQL query results in the context of Linked Data. These query result explanations are based on query result provenance. As we discuss in this chapter, existing SPARQL query result provenance computation techniques are based on annotation approaches. These approaches require re-engineering the underlying data model, the query language, and the query processing engine to compute provenance during the query processing. However, re-engineering the underlying data model, the query language, or the query processor is not an option in the Linked Data scenario. Data is hosted, served, and controlled by external parties in the Linked Data scenario. We investigate how to compute SPARQL query result provenance without re-engineering the underlying data model, the query language, or the query processor – the non-annotation approach.

Furthermore, very little has been done in the previous work in the Semantic Web literature to evaluate the validity of assumptions such as explanations would improve users' understanding and trust. We investigate how SPARQL query result explanations impact users in the context of Linked Data.

In addition, much of the previous work on explanations for the Semantic Web does not address explanation in a distributed environment. We investigate how to provide explanations for the scenario of Linked Data. In this context, the challenge is to provide explanations for distributed data produced by Linked Data applications distributed across the Web.

Finally, very few of the existing approaches address the problem of summarizing explanations. We investigate how to provide summarized explanations to provide short explanations and the ability to filter important information in explanations.

Chapter 3

Predicting Query Performance

Contents

In this chapter we address the problem of predicting SPARQL query performance. We provide the predicted performance metrics to enable users understand query behavior prior to query execution. Accurately predicting query execution time enables effective workload management (e.g. organization, inspection, and

optimization). We use machine learning techniques to learn SPARQL query performance from previously executed queries. Traditional approaches for estimating SPARQL query cost are based on statistics about the underlying data. However, in many use-cases involving querying Linked Data, statistics about the underlying data are missing. Our approach does not require any statistics about the underlying RDF data, which makes it ideal for the Linked Data scenario. We show how to model SPARQL queries as feature vectors, and use k-nearest neighbors regression and Support Vector Machine with the nu-SVR kernel to accurately predict SPARQL query execution time.

3.1 Introduction

The global data space of Linked Data presents tremendous potential for large-scale data integration over cross domain data to support a new generation of intelligent applications [Schwarte 2011]. In this context, it is increasingly important to develop efficient ways of querying Linked Data [Huang 2011]. Central to this problem is knowing how a query would behave prior to executing the query [Hartig 2007]. This enables us to adjust our queries accordingly. We present an approach to predict SPARQL query performance with the aim of assisting users (e.g. knowledge base administrators or application developers) in workload management related tasks. Knowledge base administrators can use predicted performance metrics to effectively manage workloads such that specific Quality of Service (QoS) targets are met. System architects can use query performance prediction to estimate system configurations for supporting some specific kind of workload requirements. Application developers can use query performance prediction to choose among alternative queries based on performance requirements.

Current generation of SPARQL query cost estimation approaches are based on data statistics and heuristics. Statistics-based approaches have two major drawbacks in the context of Linked Data [Tsialiamanis 2012]. First, the statistics (e.g.

histograms) about the data are often missing in the Linked Data scenario because they are expensive to generate and maintain. Second, due to the graph-based data model and schema-less nature of RDF data, what makes effective statistics for query cost estimation is unclear. Heuristics-based approaches generally do not require any knowledge of underlying data statistics. However, they are based on strong assumptions such as considering queries of certain structure less expensive than others. These assumptions may hold for some RDF data sets and may not hold for others.

We take a rather pragmatic approach to SPARQL query cost estimation. We learn SPARQL query performance metrics from already executed queries. In relation to the research questions in Section 1.2, we address the research question **RQ1**: "How to predict query performance metrics on SPARQL endpoints that provide Linked Data querying services"? Recent work [Ganapathi 2009, Gupta 2008, Akdere 2012] in database research shows that database query performance metrics can be accurately predicted without any knowledge of data statistics by applying machine learning techniques on the query logs of already executed queries. Similarly, we apply machine learning techniques to learn SPARQL query performance metrics from already executed queries. We consider query execution time as the query performance metric.

3.1.1 Publications

We published the work resulting from this chapter in the IEEE/WIC/ACM International Conference on Web Intelligence 2014 (WI 2014) as a full research paper [Hasan 2014c]; and in the Extended Semantic Web Conference 2014 (ESWC2014) as a poster [Hasan 2014d] and in a doctoral symposium paper [Hasan 2014b].

3.2 Query Performance Prediction

Recent work on predicting database query performance [Akdere 2012, Ganapathi 2009, Gupta 2008] has argued that the cost models used by the current generation query optimizers are good for comparing alternative query plans, but ineffective for predicting actual query performance metrics such as query execution time. These cost models are unable to capture the complexities of modern database systems [Akdere 2012]. To address this, database researchers have experimented with machine learning techniques to learn query performance metrics. Ganapathi *et al.* [Ganapathi 2009] use Kernel Canonical Correlation Analysis (KCCA) to predict a set of performance metrics. For the individual query elapsed time performance metric, they were able to predict within 20% of the actual query elapsed time for 85% of the test queries. Gupta *et al.* [Gupta 2008] use machine learning for predicting query execution time ranges on a data warehouse and achieve an accuracy of 80%. Akdere *et al.* [Akdere 2012] study the effectiveness of machine learning techniques for predicting query latency of static and dynamic workload scenarios. They argue that query performance prediction using machine learning is both feasible and effective.

Related to the Semantic Web query processing, SPARQL query engines can be categorized into two categories: SQL-based and RDF native query engines [Tsialiamanis 2012]. SQL-based query engines rely on relational database systems storage and query optimization techniques to efficiently evaluate SPARQL queries. They suffer from the same problems as mentioned above. Furthermore, due to the absence of schematic structure in RDF, cost-based approaches – successful in relational database systems – do not perform well in SPARQL query processing [Tsialiamanis 2012]. RDF native query engines typically use heuristics and statistics about the data for selecting efficient query execution plans [Stocker 2008]. Heuristics-based optimization techniques include exploiting syntactic and structural variations of *triple patterns* in a query [Stocker 2008], and rewriting a

query using algebraic optimization techniques [Frasincar 2004] and transformation rules [Hartig 2007]. Heuristics-based optimization techniques generally work without any knowledge of the underlying data. Stocker *et al.* [Stocker 2008] present optimization techniques with pre-computed statistics for reordering triple patterns in a SPARQL query for efficient query processing. However, in many use-cases involving querying Linked Data, statistics are missing [Tsialiamanis 2012]. This makes these statistics-based approaches ineffective in the Linked Data scenario. Furthermore, as in the case of relation database systems, these existing approaches are unable to predict actual query performance metrics such as query execution time for a given configuration.

3.3 Learning SPARQL Query Performance

We predict SPARQL query performance metrics by applying machine learning techniques on previously executed queries. We treat the SPARQL engine as a black box and learn query performance metrics from already executed queries. This approach does not require any statistics of the underlying RDF data, which makes it ideal for the Linked Data scenario. As in the common machine learning approaches, our query performance prediction approach includes two main phases: training and testing. In the training phase, we derive a prediction model from a training data set containing previously executed queries and the observed performance metric values (execution times) for those queries. We represent the queries as feature vectors. The goal of the training phase is to create an accurate model that maps the feature vectors to the performance metric data points. We use regression for this purpose. We define a feature vector as $x = (x_1, x_2, ...x_n)$, where $x \in \mathbb{R}^n$ and each x_i is a SPARQL query feature. The performance metric, query execution time, is the variable y. We learn a function $f(x) = y$, *i.e.* the function maps a feature vector x to y, using regression. We provide more details on the types of regression we use in section 3.5.3. In the testing phase, we use the trained model to predict query

performance metric values for unforeseen queries. Additionally, we tune our model parameters using cross-validation.

3.4 Modeling SPARQL Query Features

We use two types of query features: SPARQL algebra features and graph pattern features.

3.4.1 SPARQL Algebra Features

We use the frequencies of all the SPARQL algebra operators except the *SLICE* operator as query features. The *SLICE* operator is the combination of *OFFSET* and *LIMIT* SPARQL keywords. We use the sum of all the *SLICE* operator cardinalities appearing in the algebra expression as the feature representing the *SLICE* operator. In addition, we use two more features: the depth of the algebra expression tree and the number of *triple patterns*. Figure 3.1 shows an example of extracting the SPARQL algebra features vector from a SPARQL query. First we transform a query into an algebra expression tree. Then we extract the features and represent the query as a feature vector. We use the Jena ARQ SPARQL parser[1] to transform query strings to SPARQL algebra expressions.

3.4.2 Graph Pattern Features

The SPARQL algebra features do not represent graph patterns appearing in SPARQL queries. Transforming graph patterns to vectors is not trivial because the vector space is infinite. To address this, we create a query pattern vector representation relative to the query patterns appearing in the training data. First, we cluster the structurally similar query patterns in the training data into K_{gp} number of clusters. The query pattern in the center of a cluster is the representative of query patterns in that cluster. Second, we represent a query pattern as a K_{gp} dimensional

[1] https://jena.apache.org/documentation/query/algebra.html

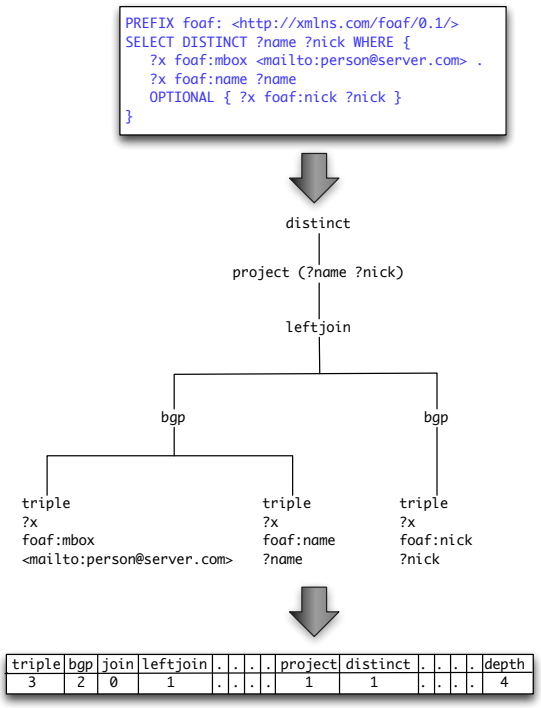


Figure 3.1: Extracting SPARQL algebra features from a SPARQL query.

vector where the value of a dimension is the structural similarity between that query pattern and the corresponding cluster center query pattern.

3.4.2.1 Structural Similarity Between Query Patterns

To compute the structural similarity between two query patterns, we first construct two graphs from the two query patterns, then compute the graph edit distance [Bunke 1994, Riesen 2009] between these two graphs. We compute the structural similarity by inverting the edit distance. To introduce the notion of graph edit distance, we paraphrase the definitions of a graph and the graph edit distance from [Riesen 2009].

Definition 4 (*Graph*) *A graph g is a tuple $g(V, E, \mu, \nu)$ where*

- *V is the finite set of nodes.*
- *$E \subseteq V \times V$ is the set of edges.*

- *L is the finite or infinite set of labels for nodes and edges.*
- *$\mu : V \rightarrow L$ is the node labeling function.*
- *$\nu : E \rightarrow L$ is the node labeling function.*

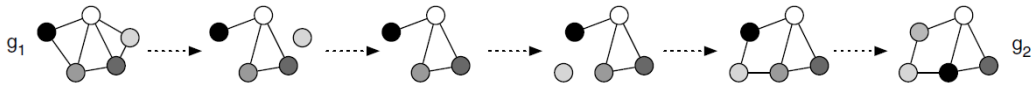


Figure 3.2: A possible edit path to transform graph g_1 to graph g_2.

The graph edit distance between two graphs is the minimum amount of distortion needed to transform one graph to another. The amount of distortion is the cost of a sequence of edit operations. Standard edit operations include deletions, insertions, and substitutions of nodes and edges. The example from [Riesen 2009] in figure 3.2 shows a possible edit path to transform graph g_1 to graph g_2. The edit operations in this path are three edge deletions, one node deletion, one node insertion, two edge insertions, and finally two node substitutions. For a pair of graphs (g_s, g_t), there can be number of edit paths to transform g_s to g_t. Let $\Upsilon(g_s, g_t)$ be the set of all such edit paths. To find the suitable edit path out of all the edit paths in $\Upsilon(g_s, g_t)$, a cost function for each edit operation is introduced. There should be an inexpensive edit path for two similar graphs, which represents low cost edit operations, while an edit path with high cost is required for two dissimilar graphs. Therefore, the edit distance of two graphs is defined by edit path with minimum cost between the two graphs.

Definition 5 (*Graph Edit Distance*) *Let $g_s(V_s, E_s, \mu_s, \nu_s)$ be the source and $g_t(V_t, E_t, \mu_t, \nu_t)$ the target graph. The graph edit distance between g_s and g_t is defined as:*

$$d(g_s, g_t) = \min_{(e_1 \ldots e_k) \in \Upsilon(g_s, g_t)} \sum_{i=1}^{k} c(e_i)$$

where $\Upsilon(g_s, g_t)$ denotes the set of edit paths for transforming g_s to g_t, and c denotes the cost function which measures the strength $c(e_i)$ of edit operation e_i.

A well known method for computing graph edit distance is using the A* search algorithm to explore the state space of possible mappings of the nodes and edges of the source graph to the nodes and edges of the target graph. However, the computational complexity of this edit distance algorithm is exponential in the number of nodes of the involved graphs, irrespective of using A* search with a heuristic function to govern the tree traversal process. Therefore we use the polynomial time suboptimal solution of graph edit distance that Riesen and Bunke [Riesen 2009, Riesen 2013] propose. The computational complexity of this polynomial time suboptimal solution is $O\left(n^3\right)$, where n is the number of nodes of the involved graphs. To construct a graph from a query pattern, we take all the *triple patterns* in the query pattern and construct a graph from these *triple patterns*. As in *RDF graphs*, the subject and the object of a *triple pattern* represent nodes of the graph and the predicate represents an edge of the graph. After constructing such a graph, we replace the labels of nodes and edges representing variables by a fixed symbol - the symbol '?'. This ensures that the graph has separate nodes and edges for each variable appearing in the query but a unified labeling. We call such a graph a *query graph*. Figure 3.3 shows an example of extracting graph pattern features for a query. First step (the upper part) shows the constructed *query graph*. For the sample query in Figure 3.3, three nodes are created for variables and one node is created for the resource `<mailto:person@server.com>`. In addition, the labels of the edges in the *query graph* are taken from the predicates of the *triple patterns*. Please note that the nodes representing a variable is always a separate note with the label '?'. For example, if there are two variables in the *triple patterns*, there will be two nodes with the label '?' for both of them (i.e. we **do not** merge all the nodes representing variables into one node). This notion is similar to the notion of blank nodes in RDF data model. The rationale behind this design choice is to keep the original struc-

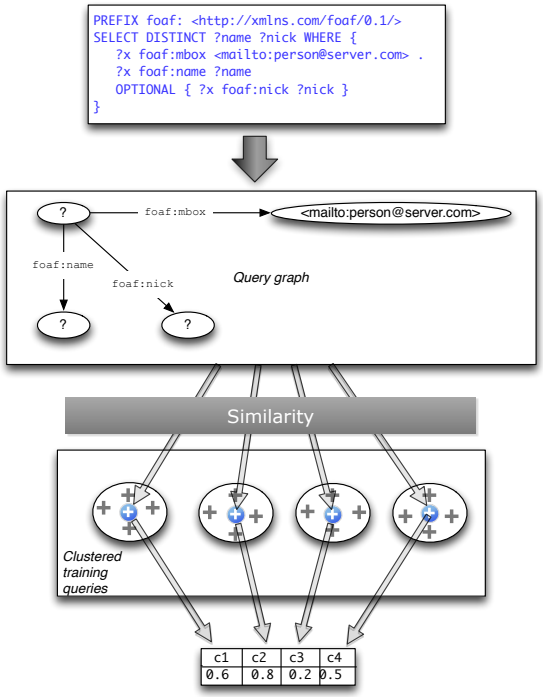


Figure 3.3: Example of extracting graph pattern features.

tures of *query graphs*, to enable us compare them, while having a unified labeling for variables. The clustered queries box in Figure 3.3 shows the clusters of training queries where each circle is a cluster of *query graphs* with their cluster centers shown in blue color.

3.4.2.2 Clustering the Training Queries

We use the k-medoids [Kaufman 1987] clustering algorithm to cluster the *query graphs* of training data. We use k-medoids because it chooses data points as cluster centers and allows using an arbitrary distance function. As we mention before, we use the suboptimal graph edit distance algorithm as the distance function for k-medoids. For the K_{gp} dimensional vector representation of query pattern, we compute the structural similarity between a *query graph* p_i and the k^{th} cluster

center *query graph* $C(k)$ as below:

$$sim(p_i, C(k)) = \frac{1}{1 + d(p_i, C(k))} \tag{3.1}$$

The term $d(p_i, C(k))$ is the graph edit distance between *query graphs* p_i and $C(k)$. This formulation gives us a similarity score within the range of 0 to 1. A similarity score of 0 being the least similar and a score of 1 being the most similar. The extracted feature vector in figure 3.3 shows the computed similarity values using equation 3.1 for the example query.

3.5 Experiments and Results

We use the DBPSB benchmark [Morsey 2011] queries on a Jena-TDB triple store [Owens 2008] to evaluate our approach. DBPSB includes 25 query templates which cover most commonly used SPARQL query features in the queries sent to DBPedia[2]. We generate our training, validation, and test queries from these query templates. We use query execution time as the query performance metric. The details of our experimental setup is described below.

3.5.1 Triple Store and Hardware

We use Jena-TDB 1.0.0 as a triple store. We allow Jena-TDB to use 16 GB of memory. We execute all the queries in a commodity server machine with a 4 core Intel Xeon 2.53 GHz CPU, 48 GB system RAM, and Linux 2.6.32 operating system.

3.5.2 Data Sets

As the RDF data set, we use the DBpedia 3.5.1 data set with 100% scaling factor – provided by the DBPSB benchmark framework. We generate our training, validation, and test queries from the 25 DBPSB query templates. To generate queries,

[2]http://dbpedia.org

we assign randomly selected RDF terms from the RDF data set to the placeholders in the query templates. We generate 205 queries for each template and then execute them to build our training, validation, and test data sets. Before executing the queries, we restart the triple store to clear the caches. Then we execute total 125 queries in our warm-up phase to measure query performance under normal operational conditions. Our warm-up queries include the first 5 queries from each of the 25 templates. To generate the training queries, we execute the next 120 queries from each template and take the first 60 queries for each template which return at least 1 result and finish executing within a reasonable time. We specify a 300 second timeout for a query execution. We follow the same process to generate 20 validation queries from the next 40 queries for each template and 20 test queries from the last 40 queries for each template. In this setting, none of the queries from template 2, 16, and 21 returned any result. All the queries from template 20 were interrupted because of timeout. This process resulted 1260 training queries, 420 validation queries, and 420 test queries. We execute each of these training, validation, and test queries 5 times and record the average execution time in milliseconds (ms) for each query. Figure 3.4 shows the average, minimum, and maximum execution times for the queries from our test data set. As the figure shows, we have a mix of long and short running queries. Queries belonging to templates 4, 10, and 24 have more than 1000 ms of average execution time. The queries from the other query templates have less than 1000 ms of average execution time.

3.5.3 Prediction Models

To predict query execution time, we experiment with two regression models. We first experiment with Weka's [Hall 2009] implementation of k-nearest neighbors (k-NN) regression [Aha 1991, Altman 1992]. The k-NN algorithm predicts based on the closest training data points. It uses a distance function to compute these closest data points. We use Euclidean distance as the distance function in our experiments. For

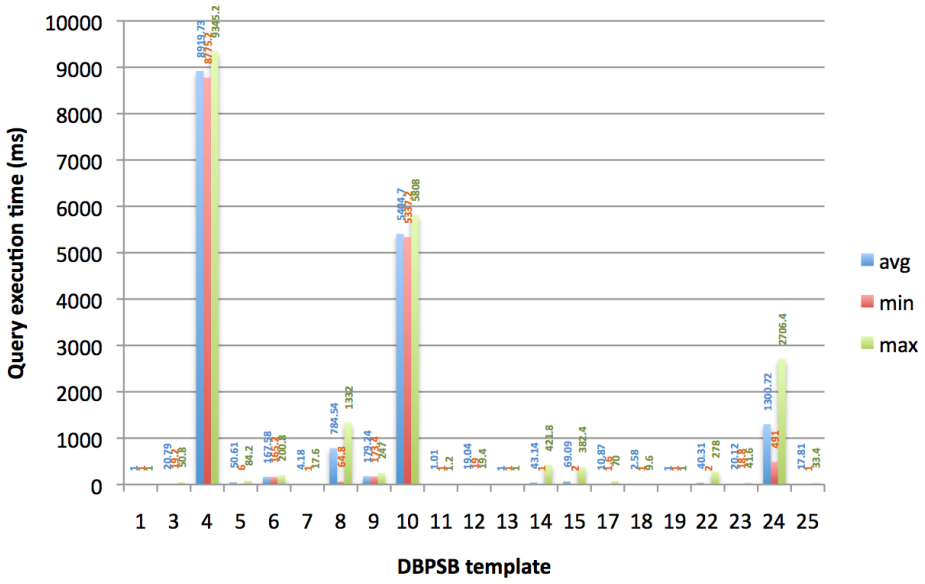


Figure 3.4: Average, minimum, and maximum execution times for the queries belonging to different query templates in the test data set.

predictions, we use the weighted average of the k nearest neighbors - weighted by the inverse of the distance from the querying data point. This ensures that the nearby neighbors contribute more to the prediction than the faraway neighbors. We use the k-dimensional tree (k-d tree) [Friedman 1977] data structure to compute the nearest neighbors. For N training samples, k-d tree can find the nearest neighbor of a data point with O ($log\, N$) operations. We also experiment with the libsvm [Chang 2011] implementation of Support Vector Machine (SVM) using the nu-SVR kernel for regression [Shevade 2000]. The approach in SVM regression is to map the features to a higher dimensional space and perform a regression in that space. The predictions in SVM are based on a subset of data points known as support vectors.

3.5.4 Evaluation Metrics

We use the coefficient of determination, denoted as R^2, to evaluate our models. R^2 is a widely used evaluation measure for regression. R^2 measures how well future

samples are likely to be predicted. We compute R^2 as:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

The vectors y and $\hat{y}$ represent the actual values and predicted values respectively for n queries. $\bar{y}$ is the mean of actual values. An R^2 score close to 1 indicates near perfect prediction. R^2 scores however can be misleading in many cases. As R^2 depends on the scale and statistical characteristics of the whole data set, it can have low errors even if the predictions have high errors [Akdere 2012]. Therefore we use another evaluation metric, root mean squared error (RMSE), as our error metric:

$$RMSE(y, \hat{y}) = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

3.5.5 Predicting Query Execution Time

We show the results of our experiments in Figure 3.5 and Figure 3.7. The results include R^2 and RMSE values using k-NN and SVM with SPARQL algebra features and graph pattern features. Below we discuss these results.

3.5.5.1 Predicting with SPARQL Algebra Features

For k-NN with SPARQL algebra features, we select k, the number of neighbors, by cross-validation. As Table 3.1 shows, different values of k do not have any effect on RMSE and R^2 on our validation data set. Therefore we select $k = 2$. We achieve an R^2 value of 0.96645 and an RMSE value of 395.5125 on the test data set using k-NN with SPARQL algebra features. Figure 3.5(a) shows the comparison between predicted and actual execution times using k-NN with SPARQL algebra features. Figure 3.5(b) shows that the queries from template 15 has the highest RMSE. The execution time for queries from template 15 range from 2 ms to 382.4 ms with an

average of 69.09 ms. Because of the high error for queries from template 15, there are overestimated data points in this interval in Figure 3.5(a).

	k=2	k=3	k=4	k=5
RMSE	588.2004	588.2004	588.2004	588.2004
R^2	0.9286	0.9286	0.9286	0.9286

Table 3.1: RMSE and R^2 values for different k for k-NN on the validation data set.

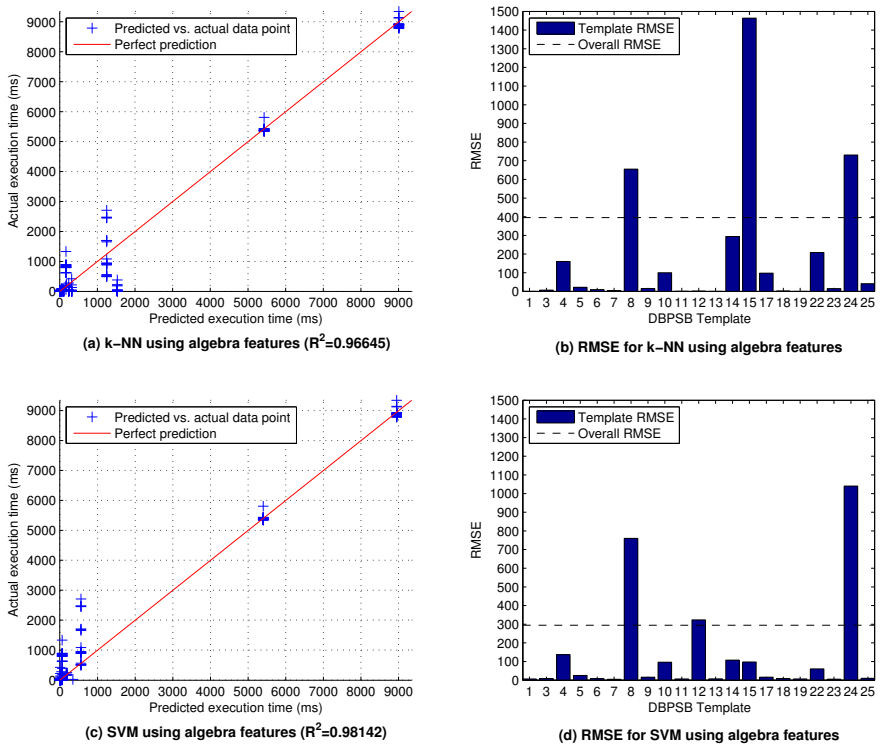


Figure 3.5: Query execution time predictions with SPARQL algebra features using k-NN (with $k = 2$) and SVM models.

We achieve an improved R^2 value of 0.98142 and a lower RMSE value of 294.3532 on the test data set using SVM with SPARQL algebra features. Figure 3.5(c) shows the comparison between predicted and actual execution times using SVM with SPARQL algebra features. Figure 3.5(d) shows the RMSE values by query template for this model. As the figures show, the error for queries from template 15

decreases. Therefore the overestimated data points in the interval 2 ms to 382.4 ms move towards the perfect prediction line. However, the error for template 8 and 24 slightly increases.

3.5.5.2 Predicting with SPARQL Algebra and Graph Pattern Features

For k-NN with SPARQL algebra features and graph pattern features, we have two parameters: the number of clusters K_{gp} and the number of neighbors k. Again we select them by cross-validation. Figure 3.6(a) shows the RMSE values on the validation data set for different K_{gp} and k, and Figure 3.6(b) shows the R^2 values on the validation data set for different K_{gp} and k. The Figure 3.6 shows, k again does not have any impact. We get lowest K_{gp} and highest R^2 values at $K_{gp} = 10$ and $K_{gp} = 20$ for all k values. Therefore we select $K_{gp} = 10$ and $k = 2$ for our predictions with k-NN on the test data set. Figure 3.7(a) and Figure 3.7(b) shows

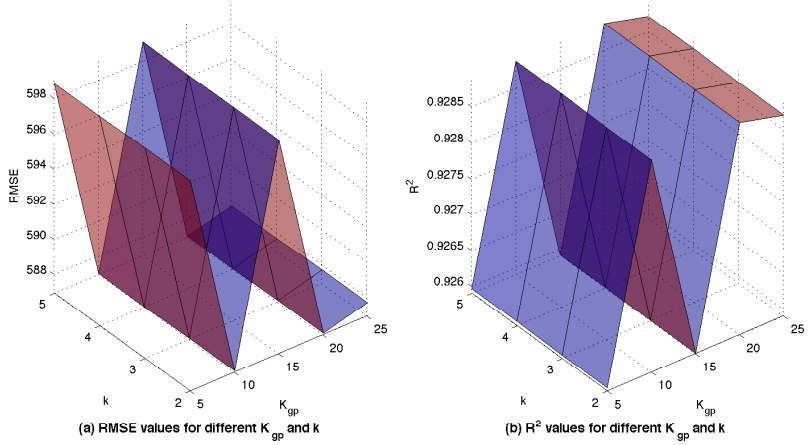


Figure 3.6: RMSE and R^2 values on the validation data set for different K_{gp} and k.

the prediction results on the test data set using k-NN with $K_{gp} = 10$ and $k = 2$. We get a slightly less R^2 value for this model than k-NN with SPARQL algebra features. This is because of the increase in RMSE values for queries from template 9, 17, and 24.

For SVM with SPARQL algebra features and graph pattern features, we select

the value of K_{gp} by cross-validation. Table 3.2 shows RMSE and R^2 values on the validation data set for different K_{gp} using SVM. We select $K_{gp} = 25$ because it gives us the lowest RMSE value 528.9321 and highest R^2 value 0.9422 on the validation data set. Figure 3.7(c) and Figure 3.7(d) shows the prediction results on the test data set using SVM with $K_{gp} = 25$. We get the overall best R^2 value 0.98526 and

	K_{gp}=5	K_{gp}=10	K_{gp}=15	K_{gp}=20	K_{gp}=25
RMSE	530.9169	546.7406	547.6764	547.4219	528.9321
R^2	0.9418	0.9383	0.9381	0.9381	0.9422

Table 3.2: RMSE and R^2 values on the validation data set for different K_{gp} using SVM.

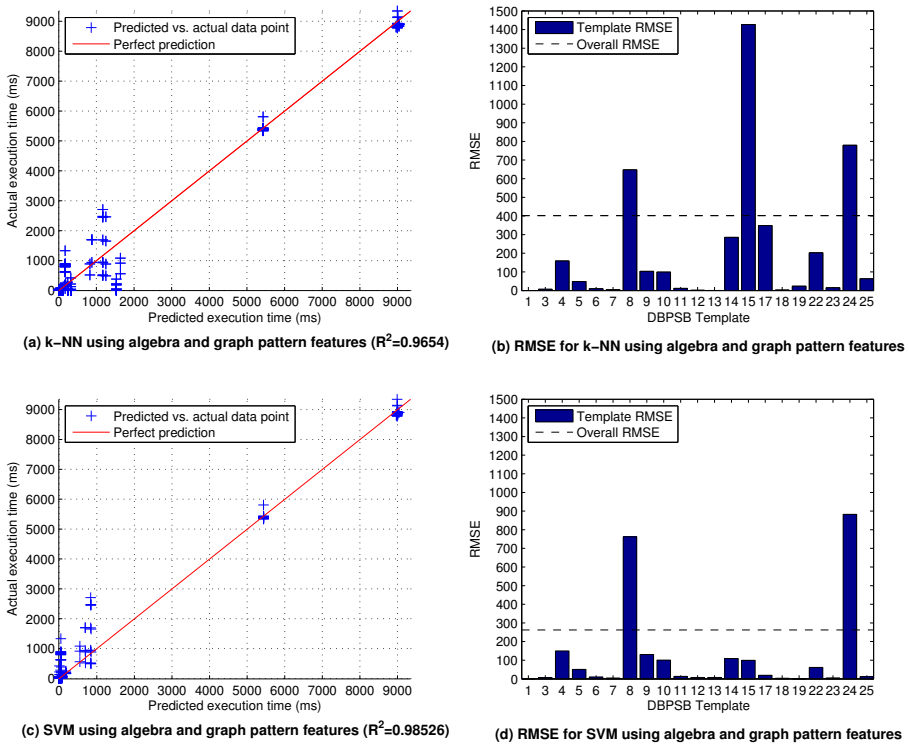


Figure 3.7: Query execution time predictions with SPARQL algebra features and graph pattern features using k-NN ($K_{gp} = 10$ and $k = 2$) and SVM ($K_{gp} = 25$).

the overall lowest RMSE value 262.1869 with this model. This is an improvement from the SVM with SPARQL algebra features model. The main reason for this is

the decrease in RMSE for queries from template 12 and 24.

3.5.6 Required Time for Training and Prediction

Table 3.3 shows the total training time and average prediction time per query for the models we experimented with. Models with SPARQL algebra features take very low prediction time per query. Training time is also low. Models with graph pattern features take longer time to train. This is because the training time includes generating the distance matrix using approximated graph edit distance. This process itself takes 3293 seconds on average for 1260 queries. Also it includes the time required to cluster the training queries. However the average prediction time per query using models with graph pattern features is within 100 milliseconds, which is reasonable especially for query solving over Linked Data. The average prediction

Model	Training time	Avg. prediction time per query
k-NN + algebra	7.14 sec	3.42 ms
SVM+ algebra	26.26 sec	3.53 ms
k-NN + algebra + graph pattern	3300.33 sec	47.25 ms
SVM + algebra + graph pattern	3390.71 sec	98.1 ms

Table 3.3: Required time for training and predictions.

time per query using models with graph pattern features increase from the models with only algebra features because of the similarity computations using approximated edit distance. It is important to note that the training phase is an offline process and hence it does not influence query prediction time.

3.6 Summary

In this chapter, we discussed assisting users in understanding query behavior. We presented a machine learning approach to SPARQL query performance prediction. We learn query execution times from already executed queries. This approach can

be useful where statistics about the underlying data are unavailable – the Linked Data scenario. We discuss how to model SPARQL queries as feature vectors, and show highly accurate predictions. Users such as knowledge base administrators or application developers, in the Linked Data scenario, can use the predicted performance metrics using our approach to effectively manage workloads such that specific Quality of Service (QoS) targets are met.

In the next chapter, we discuss assisting users in understanding query results in the context of Linked Data.

Chapter 4

Explaining SPARQL Query Results

Contents

In the previous chapter, we discussed assisting users in understanding query behavior. In this chapter, we discuss assisting users in understanding query results. We present an approach to explain SPARQL query results. We generate the explanation for a query result tuple from its *why-provenance*. We present a non-annotation approach to generate *why-provenance* and show its feasibility for Linked Data. We present an explanation-aware federated query processor prototype and show the presentation of our explanations.

4.1 Introduction

As argued in [Theoharis 2011, Wylot 2014], it is essential to provide additional explanations about which source data were used in providing results, how the source data were combined, to enable users understand the result derivations, and validate or invalidate the results.

Within the Semantic Web community, explanations have been studied for Semantic Web applications and OWL entailments. Explanation for SPARQL query results has not been independently studied by the community. However, there have been several works on tracing the origin of query results – e.g. *why-provenance*. These attempts are based on what is known as the annotation approach (the eager approach) where the underlying data model, the query language, and the query processing engine are re-engineered to compute provenance during the query processing. This is undesirable for the Linked Data scenario as re-engineering the underlying data model, the query language, or the query processor is often not possible from the querying side.

In this chapter, we address the research question **RQ2**: "How to provide explanations for SPARQL query results on SPARQL endpoints that provide Linked Data querying services"? We propose a non-annotation approach to generate *why-provenance* for SPARQL query results. We generate the explanation for a query result tuple from its *why-provenance*. We generate *why-provenance* of SPARQL query results without modifying the RDF data model, the query language, or the query processor. Our approach is suitable for scenarios where querying clients are required to generate provenance from the querying side and are not allowed to modify the query processor or the underlying data model – the Linked Data scenario. Additionally, provenance metadata is generated only when it is needed – commonly known as the lazy approach. Therefore, our approach does not have any query execution time overhead or provenance metadata storage overhead. Finally, we present an explanation-aware federated query processor prototype to show the presentation

of our explanations.

4.1.1 Publication

We published the results of this chapter in a full research paper [Hasan 2014e] in the Semantic Web Collaborative Spaces Workshop 2014 (SWCS 2014) at the 13th International Semantic Web Conference (ISWC 2014).

4.2 Explanation and Provenance

As we discuss in Chapter 2, previous work on explanation in the Semantic Web literature addresses the problems of representing explanation metadata [Pinheiro da Silva 2006], and generating explanations for Semantic Web applications [McGuinness 2008] and entailments [Horridge 2008]. SPARQL query result explanation has not be independently studied in the previous work. However, query result provenance has been studied in the database community [Cheney 2009] and the Semantic Web community. Table 4.1 shows a comparison of query result provenance approaches in the Semantic Web literature.

	[Theoharis 2011]	[Damásio 2012]	[Wylot 2014]	[Corby 2012]	[Dividino 2009]	[Zimmermann 2012]
Annotation/eager	√	√	√	√	√	√
Non-annotation/lazy						
Data model transformation	√	√			√	√
Query language transformation	√	√				

Table 4.1: Comparison of query result provenance approaches in the Semantic Web.

The previous works on provenance for SPARQL query results are based on transforming the RDF data model and SPARQL query language to relational data model and relational database query language respectively and then applying relational

database approaches [Theoharis 2011, Damásio 2012]; transforming the original data model to annotated RDF or named graphs [Dividino 2009, Zimmermann 2012]; or generation of provenance metadata during the query processing (annotation or eager approach) [Theoharis 2011, Damásio 2012, Wylot 2014, Corby 2012, Dividino 2009, Zimmermann 2012]. However, we do not have any control over the underlying data model or the query processor in the Linked Data scenario. Therefore, re-engineering the underlying data model or query processor is often not possible in the Linked Data scenario. In this context, we need an approach which can be deployed without re-engineering the underlying system. This is a perfectly suitable scenario for the non-annotation approach. The non-annotation approaches for relational databasesare not applicable in this scenario because one has to first transform the RDF data to relational data and the queries to relational database queries to use those non-annotation approaches.

4.3 Explaining SPARQL Query Results

We provide SPARQL query result provenance as query result explanations. More precisely, for a SPARQL query result tuple, we provide its *why-provenance* as its explanation. Buneman *et al.* [Buneman 2001] first introduced the notion of *why-provenance* for relational databases. *Why-provenance* captures all the different witnesses for a tuple in the query result. For a query Q and output tuple t, a *witness* is the sufficient subset of the database records which ensures that the tuple t is in the output. Each witness is a derivation for the output tuple. Theoharis *et al.* [Theoharis 2011] later adapted *why-provenance* for RDF and SPARQL. Similar to the relational setting, *why-provenance* for RDF and SPARQL captures all the different derivations of a tuple in the query result. To illustrate, we use a simple example, containing RDF data about professors and the courses they teach, shown in Figure 4.1. We use identifiers for each triple for presentation purpose in this chapter. Consider the SPARQL query *Q1* shown in Listing 4.1, which asks for all

	Triples about professors
t1	:ProfA :dept :CS
t2	:ProfA :name "Prof. A"
t3	:ProfA :email "a@email.edu"
t4	:ProfA :course :CS101
t5	:ProfA :course :CS103
t6	:ProfA :course :CS201
t7	:ProfA :course :CS204
t8	:ProfB :dept :MATH
t9	:ProfB :name "Prof. B"
t10	:ProfB :email "b@email.edu"
t11	:ProfB :course :MATH101
t12	:ProfB :course :MATH201

	Triples about courses
t13	:CS101 :courseType :underGrad
t14	:CS103 :courseType :underGrad
t15	:MATH101 :courseType :underGrad
t16	:CS201 :courseType :grad
t17	:CS204 :courseType :grad
t18	:MATH201 :courseType :grad

Figure 4.1: Example RDF triples.

the professors who teach undergraduate level courses and their corresponding email addresses. The first triple pattern *?course :courseType :underGrad* (line 3) selects the undergraduate level courses.

Listing 4.1: SPARQL query Q1

```
1 SELECT  DISTINCT ?name ?email
2 WHERE
3   { ?course :courseType :underGrad .
4     ?prof :course ?course .
5     ?prof :email ?email .
6     ?prof :name ?name
7   }
```

Result of Q1:

?name	?email
Prof. A	a@email.edu
Prof. B	b@email.edu

The second triple pattern *?prof :course ?course* (line 4) selects the professors for those undergraduate level courses. The next two triple patterns *?prof :email ?email* (line 5) and *?prof :name ?name* (line 6) selects the email addresses and names of the corresponding professors matched by the two previous triple patterns. The result of the query *Q1* (under set semantics) executed on the RDF data containing the triples in Figure 4.1 is shown on the right in Listing 4.1. The *why-provenance* for the result tuple (Prof. A, a@email.edu) is {{t14, t5, t2, t3}, {t13, t4, t2, t3}}. Each inner set in *why-provenance* represents a derivation involving the triples in the inner set. This means that the result tuple (Prof. A, a@email.edu) can be derived in two different ways according to *Q1*. The first one by using the triples t14, t5, t2, and t3.

The second one by using the triples t13, t4, t2, and t3. The *why-provenance* for the result tuple (Prof. B, b@email.edu) on the other hand has one derivation: {{t15, t11, t10, t9}}. Please note that we are using the triple identifiers only for presentation purpose. The original data model containing the triples shown in Figure 4.1 is not changed – *i.e.* we do not annotate the RDF triples. We use the RDF triples as they are in the original data source.

4.3.1 Algorithm for Generating Why-Provenance

In this section, we present our non-annotation approach to generate *why-provenance* for SPARQL query results. We currently do not support SPARQL queries with subqueries, FILTER (NOT) EXISTS, MINUS, property paths, and aggregates. The *GenerateWhyProvenace* procedure shown in Algorithm 1 generates *why-provenance* for an RDF model M, a SPARQL query Q, and a result tuple t. The RDF model M can be an RDF data set or a SPARQL endpoint on which the SPARQL query Q is solved and the result tuple t is produced. At line 2 of Algorithm 1, we first

Algorithm 1 Why-provenance algorithm.

```
1: procedure GenerateWhyProvenace(M,Q,t)
2:     Q' ← ProvenanceQuery(Q, t)
3:     I ← Q'(M)
4:     E ← AlgebraicExpression(Q)
5:     W ← DerivationsFromQuery(M, E, I)
6:     return W
```

re-write the original query to a provenance query by adding the tuple t as a solution binding using the SPARQL 1.1 VALUES construct, and projecting all the variables. The result set of the provenance query provides us with all the variable bindings on the RDF data for the solution tuple t. Each tuple (row) in the result set of the provenance query represents a derivation for the solution tuple t. The main idea behind our algorithm is to extract *why-provenance* triples from the triple patterns in the original query by replacing the variables in the triple patterns by the corresponding values from each tuple (row) of result of the provenance query.

At line 3 of Algorithm 1, we execute the re-written query. At line 4, we convert the original SPARQL query Q to SPARQL algebraic expression for ease of query parsing and manipulation. At line 5, the *DerivationsFromQuery* procedure extracts the derivations by iterating through all the tuples of the provenance query result and replacing the variables of triple patterns in the original query by the corresponding values in a tuple of the provenance query result.

Listing 4.2: Provenance query Q2

```
1  SELECT  *
2  WHERE
3    { ?course :courseType :underGrad .
4      ?prof :course ?course .
5      ?prof :email ?email .
6      ?prof :name ?name
7    }
8  VALUES ( ?email ?name ) {
9    ( "a@email.edu" "Prof. A" )
10 }
```

Result of Q2:

?course	?prof	?email	?name
:CS103	:ProfA	a@email.edu	Prof. A
:CS101	:ProfA	a@email.edu	Prof. A

For example, query *Q1* shown in Listing 4.1 for the result tuple (Prof. A, a@email.edu), is re-written to query *Q2* shown in Listing 4.2. The result of *Q2*, shown in the bottom of Listing 4.2, provides us with all the variable bindings on the RDF data for the solution tuple (Prof. A, a@email.edu). Each tuple (row) in this result set represents a derivation for the solution tuple.

Algorithm 2 shows the *ProvenanceQuery* procedure to re-write the original query to a provenance query. Line 2 adds the result tuple t as a solution binding using the SPARQL 1.1 VALUES construct. Line 3 modifies the query to projects all the

Algorithm 2 Procedure for creating the provenance query.

```
1: procedure ProvenanceQuery(Q,t)
2:     Q' ← AddValueBindings(Q', t)
3:     Q'' ← ProjectAllVariables(Q')
4:     return Q''
```

variables in the query.

Algorithm 3 Procedure for extracting derivations from a query.

```
1: procedure DerivationsFromQuery(M,E,I)
2:     D ← ∅
3:     for each tuple in I do
4:         for each bgp in E do
5:             BP[bgp] ← False
6:         T ← ∅
7:         if hasUnion(E) or hasJoin(E) or hasLeftJoin(E) then
8:             for each operator in E do
9:                 T1 ← TriplesForOperator(M, operator, tuple, BP)
10:                if T1 ≠ ∅ then
11:                    T ← T ∪ T1
12:        else
13:            bgp ← GetTheBGP(E)
14:            T ← TriplesFromBGP(M, bgp, tuple, BP)
15:        D ← D ∪ {T}
16:    return D
```

Algorithm 3 shows the *DerivationsFromQuery* procedure to extract the derivations given the RDF model M, the SPARQL algebraic expression E, and the provenance query results I. Lines 3–15 iterate through all the tuples of I, extract provenance triples corresponding to each tuple, and store them in a set of a sets D. We assume that no basic graph pattern (BGP) is repeated in the SPARQL query. We use a hash table, BP, to flag which BGP is examined for a tuple in I to extract provenance triples. Lines 4–5 initialize the hash table by setting the value of each BPG to *False*, meaning none of the basic graph patterns is examined for the current tuple in I at this point. If a query has just one BGP, we extract the provenance triples from that BGP (lines 13–14) for a tuple in I and store the provenance triples in set T. If a query has more than one BGP, *i.e.* if the algebraic expression has

the union operator or the join operator or the left-join operator, we extract the provenance triples from the operand BGPs of each of the operators and store the provenance triples in set T (lines 8–11) for a tuple in I. We only extract provenance triples for a BGP once at this stage – using the hash table BP as flags for BGPs to keep trace of which BGP has been used so far to extract provenance triples. Finally line 15 does a union of the triples extracted for a tuple in I, stored in set T, as an element (shown by braces around T at line 15) with the set of sets D and assigns the result of the union to D. When we exit the loop started at line 3, D contains all the derivations we extracted. We return the set of sets D at line 16. Each element in D is a set representing a derivation for the result tuple.

Algorithm 4 Procedure for extracting triples from operands of an operator.

```
1:  procedure TriplesForOperator(M, Op, Tup, BP)
2:      P ← ∅
3:      L ← GetLeftBGP(Op)
4:      R ← GetRightBGP(Op)
5:      if BP[L] = False then
6:          P ← TriplesFromBGP(M, L, Tup, BP)
7:      if BP[R] = False then
8:          T ← TriplesFromBGP(M, R, Tup, BP)
9:          P ← P ∪ T
10:     return P
```

Algorithm 4 shows the *TriplesForOperator* procedure which extracts provenance triples from the operands of an operator. Lines 3–4 get the left and the right BGPs for the operator Op. As we are restricted to SPARQL queries without sub-queries, the operands are always BGPs. Lines 5–6 extract provenance triples from the left BGP L if provenance triples have not been extracted from L yet, and assigns them to the set P. Lines 7–9 extract provenance triples from the right BGP R, stored in the set T, if provenance triples have not been extracted from R yet, and assigns the union of P and T to P. At line 10, we return the set P which contains all the provenance triples extracted from the left and the right BGPs of the operator Op. The *TriplesFromBGP* procedure calls at line 6 and line 8 check if all the

triples extracted from the BGPs exist in the RDF model M by sending SPARQL ASK queries with each extracted triples. This means that a BGP which was an operand of a SPARQL UNION or OPTIONAL operator would contribute to the provenance triples only if it matches against the RDF model M. Algorithm 5 shows the *TriplesFromBGP* procedure which does this. Lines 3–9 iterate through the triple

Algorithm 5 Procedure for extracting triples from a basic graph patter.

```
1:  procedure TriplesFromBGP(M,BGP,Tup,BP)
2:      T ← ∅
3:      for each triplePattern in BGP do
4:          triple ← ReplaceVariablesByValues(triplePattern, Tup)
5:          if Ask(M, triple) = True then
6:              T ← T ∪ triple
7:          else
8:              BP[BGP] ← True
9:              return ∅
10:     BP[BGP] ← True
11:     return T
```

patterns in the BGP and extracts the triples. At line 4 we replace the variables of a triple pattern by the corresponding values in the tuple *Tup*, where *Tup* is a tuple from the result of the re-written provenance query. Lines 5–6 first check if the extracted triple is valid by sending an ASK query with this triple to the RDF model M, then if it's a valid triple we take the triple and store it in the set T. If the triple is not valid (does not exist in M), we set the flag for the BGP to true and return an empty set (lines 7–9). At line 10, we exit the loop started at line 3, and set the flag for the BGP to true. Finally at line 11 we return the set of extracted provenance triples.

4.4 Performance Evaluation of the *Why-Provenance* Algorithm

We implemented our *why-provenance* algorithm using Jena-ARQ API[1]. We evaluated our algorithm using the DBPSB benchmark [Morsey 2011] queries on a Jena-TDB (version 1.0.0) triple store [Owens 2008]. DBPSB includes 25 query templates which cover most commonly used SPARQL query features in the queries sent to DBPedia[2]. We generated our benchmark queries from these query templates. We allowed Jena-TDB to use 16 GB of memory. We executed all the queries in a commodity server machine with a 4 core Intel Xeon 2.53 GHz CPU, 48 GB system RAM, and Linux 2.6.32 operating system. As for the RDF data set, we used the DBpedia 3.5.1 data set with 100% scaling factor – provided by the DBPSB benchmark framework. To generate benchmark queries, we assigned randomly selected RDF terms from the RDF data set to the placeholders in the DBPSB query templates. We generated 1 query for each template resulting in a total 25 queries. Before executing the queries, we restarted the triple store to clear the caches. Then we executed the 25 queries along with the *why-provenance* algorithm for all the result tuples once in the warm-up phase. Then we executed each query and the *why-provenance* algorithm for all the result tuples of each query 5 times. We report the average execution time and average provenance generation time for all result tuples (PGT) for each query, both in milliseconds (ms). We specify a 300 second timeout for a query execution. Queries belonging to templates 2, 16, 20, and 21 did not finish executing within the 300 seconds time limit, and hence we do not report them.

4.4.1 Query Execution and Provenance Generation

Table 4.2 shows the number of results (#RES), query executing time (QET), provenance generation time for all result tuples (PGT), provenance generation overhead

[1] http://jena.apache.org/
[2] http://dbpedia.org

in percentage for all results (PGO), and provenance generation time per result tuple (PGTPR) for DBPSB queries. PGTs for queries with long execution times and large number of results (queries 6, 8, 10, 14, 22, 24, and 25) are very high. This is not surprising because for each result tuple of a query, we execute the original query with the result tuple as a variable-value binding. Database literature already discusses this issue [Cheney 2009]. Generally speaking, non-annotation approaches compute provenance only when it is needed, by examining the source data and the output data. This requires sophisticated computations involving the source data and the output data. This means each individual tuple in the output data has to be examined separately to compute its provenance, and hence time required for generating provenance for all the result tuples for a query is high. Therefore the overhead of tracking provenance for all result tuples (PGO) in our experiment is as high as 61587.16% (query 25). Non-annotation approaches are effective in scenarios where provenance is required for a selected number of result tuples of an already solved query. Hence considering the original query execution time or the provenance generation time for all result tuples is not required. In contrast to the annotation approaches (as in [Wylot 2014]), non-annotation approaches (such as our approach) do not affect the query processing time. Our scenario of providing query result explanations is suitable for the non-annotation approach. We only need provenance for the result tuple for which the explanation is asked. Therefore, provenance generation time per result tuple (PGTPR) is the interesting measure for us. PGTPR for all the queries are low, ranging from 0.001 ms to 85.8 ms. Even for the long running queries, PGTPR values are low. This is because we add the variable-value binding to the original query to compute provenance, which makes the query simpler to solve for the query processor. This experiment shows that our algorithm is suitable for practical queries on Linked Data to generate *why-provenance* for single result tuples.

Query	#RES	QET (ms)	PGT (ms)	PGO (%)	PGTPR (ms)
1	4	25	12.2	48.8	3.05
3	1	75	65.6	87.47	65.6
4	2	8495.6	8.4	0.099	4.2
5	13	78	102.6	131.54	7.89
6	3238	785	428.2	54.55	0.13
7	21	4.2	57.8	1376.2	2.75
8	60447	7392.4	1035.4	14.01	0.017
9	4	1156.2	341.2	29.51	85.3
10	2933	6506.8	164828	2533.17	56.2
11	1	0.4	0.01	2.5	0.01
12	1	18.4	43.8	238.043	43.8
13	2	0.4	0.4	100	0.2
14	4137	604.6	7999.6	1323.123	1.93
15	38	925.6	0.2	0.022	0.005
17	82	20.6	0.6	2.913	0.007
18	34	0.6	0.2	33.333	0.006
19	2	0.4	0.002	0.5	0.001
22	82298	7424.4	405456.4	5461.134	4.927
23	1	16.6	17.8	107.229	17.8
24	134968	5729	1700.4	29.681	0.013
25	47696	1683.4	1036758.2	61587.157	21.737

Table 4.2: Query execution and provenance generation times for DBPSB queries.

4.5 An Explanation-Aware Federated Query Processor Prototype

So far we have discussed generating *why-provenance* for SPARQL query results. In this section we discuss how we use *why-provenance* to provide explanations in the context of querying and data integration over Linked Data. As we discuss in Section 2.3, a large number of Linked Data publishers provide SPARQL endpoints for directly querying the data. Query federation is a prominent approach to consume, process, and integrate Linked Data. We present a prototype system for federated query processing with explanation features. Users can ask for explanation for each query result tuple in our system. We implement a virtual integration-based federated query processor. The first step for our federated query processing is selecting the data sources by sending SPARQL ASK queries with each triple pattern. Next, we split the original query to sub-queries, sequentially send them to the relevant data sources (nested loop join), and combine the result in the local federator. Each sub-query is a CONSTRUCT SPARQL query which returns a set of matched triples for its triple patterns. We create a local virtual graph combining the resulted triples from all the sub-queries, then locally solve the original query on this virtual graph using Jena-ARQ. We borrow the idea of CONSTRUCT sub-queries from Corese-DQP [Gaignard 2013]. We also implement the common federated query processing concepts of exclusive triple pattern groups and bound join proposed in [Schwarte 2011]. The objective of exclusive grouping is to group together triple patterns which can be solved in the same data source, so that sub-queries with a group of triple patterns can be sent to the data sources instead of sending sub-queries of each individual triple pattern. Bound join replaces the variables in the sub-queries by corresponding values from previously solved sub-queries in the nested loop join. This reduces the amount of results for sub-queries.

We provide a user interface to enable users to configure SPARQL endpoints as data sources, and submit queries. Figure 4.2 shows the querying user interface

Figure 4.2: User interface for submitting queries. Users can write a SPARQL query in this user interfaces (as show in the upper part of the user interface), then click the "Query" button to solve the query. After the query is solved, each result tuple appears with a "Explain" button. Users can click the "Explain" button of a result tuple to ask for its explanation.

of our prototype. Users can ask for explanation for each query result tuple from this user interface. We provide three types of information in an explanation. We show the triples for the first derivation from the *why-provenance*, which data source each triple in the derivation comes from, and which triple pattern of the original query each triple in the derivation matches. Figure 4.3 shows an example of a query result explanation. We generate the *why-provenance* triples using the algorithm we presented in section 4.3.1 on the local virtual RDF graph. We keep two additional indexes in the federated query processor to keep tack of which data source each triple comes from, and which triple pattern each triple matches. These two indexes allow us to provide the information on data sources and matched triple patterns in the explanations.

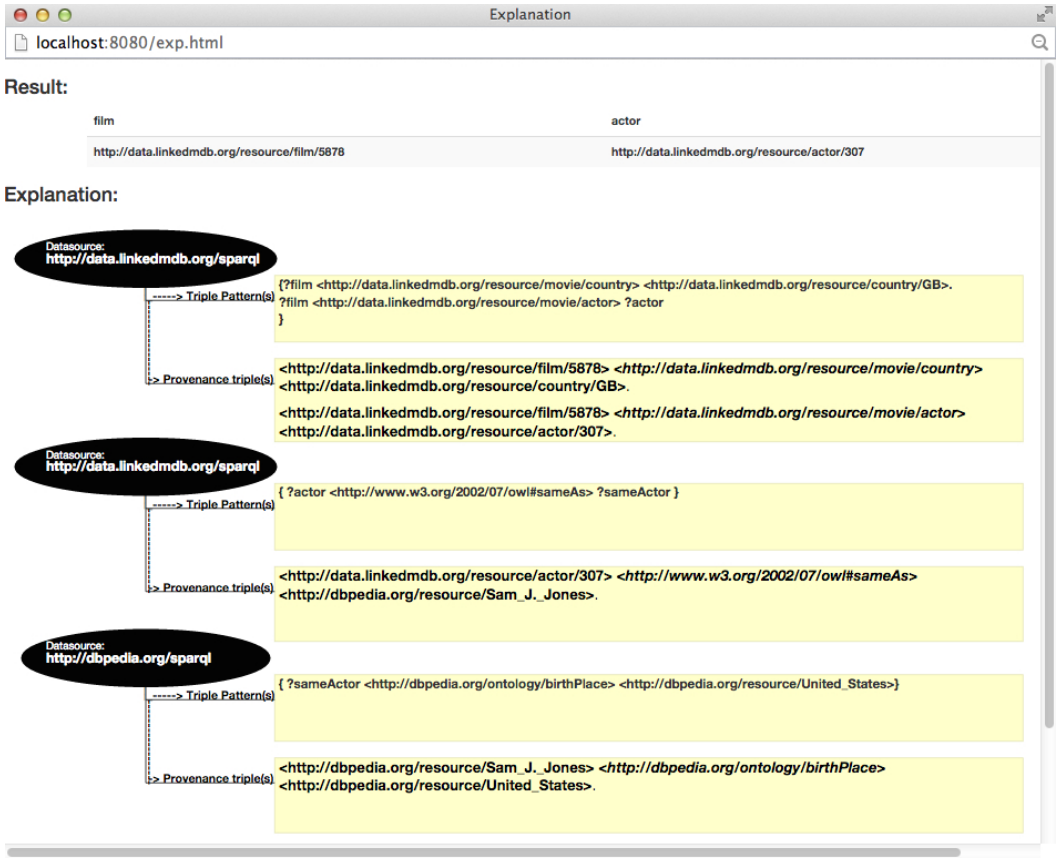


Figure 4.3: Example of a query result explanation. First we present the result tuple that the explanation user interface is explaining. Each of the oval shaped nodes presents the URL of a SPARQL endpoint. Each oval shaped node is connected to two box shapes (shown in yellow). The first box presents the triple pattern(s) which are matched in the corresponding SPARQL endpoint. The second box presents the triple(s) which are matched in the corresponding SPARQL endpoint – the first derivation of the *why-provenance* for the result tuple.

4.6 Summary

In this chapter, we discussed assisting users in understanding query results in the context of Linked Data. We provide SPARQL query result explanations to help users in understanding query result derivations. We generate query result explanations from *why-provenance* for query results. We presented a non-annotation approach to generate SPARQL query result provenance. Our non-annotation approach allows to generate *why provenance* without the RDF data model, the query language, or the query processor – which is the case in querying Linked Data. We show the feasibility of our approach for common Linked Data queries. Finally, we discuss how we use our *why-provenance* approach to provide query result explanations in the scenario of federated query processing over Linked Data.

In the next chapter, we present a user study to investigate how the query result explanations impact users.

Chapter 5

Impact of Query Result Explanations

Contents

In the previous chapter we discussed how we provide SPARQL query result explanations in a federated query processing scenario. In this chapter we present a user study to evaluate the impact of the query result explanations. Our study shows that our query result explanations are helpful for users to understand the result derivations and make trust judgments on the results.

5.1 Introduction

Much of the previous work on explanations in the Semantic Web literature has focused on representation and generation of explanations. As McGuinness *et al.* [McGuinness 2003, McGuinness 2004] discuss, explanations are provided to help users improve their understanding of the process of deriving results and the flow of information involved in the process. The improved understanding may lead to better user acceptance, and hence improved trust on the Semantic Web applications. These values of explanations have however not been evaluated in the Semantic Web literature.

In this chapter, we present a user study which evaluates the impact of query result explanations in the scenario of federated query processing over Linked Data. This relates to the research question **RQ3**: "what are the impacts of query result explanations"? In particular, we study whether providing explanations for federated query results improve users' understanding of the query solving process, and help them make trust judgments on the results. Federated query processors first split a query into sub-queries, then solve the sub-queries in the relevant data sources (SPARQL endpoints), and finally integrate the results of the sub-queries to provide the results for the original query. In this scenario, a user may want to know which data sources contributed to the results or which part of the original query was solved with which data source. We provide explanations to help users understand these aspects of a query solution – using our explanation-aware federated query processor prototype presented in Chapter 4 – and evaluate whether the explanations help users to understand the query solving process, and to make trust judgments on the query results.

5.1.1 Publication

We published the results of this chapter in a full research paper [Hasan 2014e] in the Semantic Web Collaborative Spaces Workshop 2014 (SWCS 2014) at the 13th

International Semantic Web Conference (ISWC 2014).

5.2 Evaluating Explanations

Very little has been done to evaluate how explanations impact the users of Semantic Web applications. Silva *et al.* [Pinheiro da Silva 2008] present a user study to verify if explanations play a role for scientists to understand uncertainties related to geospatial information. Their study shows that the accuracy and the confidence in determining the quality of geospatial information (maps) significantly improved when the scientists were provided with explanations.

In other fields, Tintarev and Masthoff [Tintarev 2012] studied the effectiveness of explanations for recommender systems. The authors present user studies in two domains investigating the impact of personalization and feature-based explanations on effectiveness (helping users to make good decisions) and satisfaction (the ease of use or enjoyment). The authors found that personalization increased satisfaction, but it was harmful for effectiveness. Lim *et al.* [Lim 2009] studied the impact of explanations on end-users for context-aware applications. The authors present a controlled study comparing four different types of explanations: why, why not, how to, and what if. The authors found that providing explanations for context-aware applications to novice users – in particular Why explanations – improves users' understanding and trust in the system. Our work is in the same line as the work of Lim *et al.*. We also investigate users' understanding and trust, but for the scenario of federated query processing over Linked Data.

5.3 Impact of Query Result Explanations

Based on the requirements and the assumptions presented in the previous work on explanations for the Semantic Web (presented in Chapter 2), we hypothesize that explanations would improve user experience, where we define user experience as

the users' understanding of the system and their perception of trust on the results. Therefore we expect:

H1. Query result explanations improve user experience over having no explanations.

To test this hypothesis, we conducted a user study that investigates the impact of query result explanations. Our study is similar to the user study conducted by Lim *et al.* [Lim 2009] to examine the impact of explanations for context-aware intelligent systems. We describe our user study next.

5.3.1 Method

The questionnaire for our study consists of three sections: learning section, reasoning section, and survey section. Furthermore, we have two cases: with explanation and without explanation. A participant is randomly assigned to one of those two cases.

In the learning section, participants were given a high-level overview of our query processor and an example SPARQL query with a result tuple to help them learn how the federated query processor works. Participants for the "with explanation" case additionally received the explanation of the result tuple for the example query (as shown in Figure 4.3).

In the reasoning section, participants were given the same SPARQL query as in the learning section, but a different result tuple along with the some triples contained in two data sources (DBpedia[1] and LinkedMDB[2]). Then we first asked the participants to select the relevant data sources for each triple pattern in the query. Next, we asked the participants to select the source triples (*why-provenance* triples) from the two data sources which contributed to the result tuple. Then we asked the participants to rate their confidence on their answer choices for the data source selection and the source triple selection questions. The choices for confidence rating were very low, low, medium, high, and very high. The questions in the

[1] http://dbpedia.org/
[2] http://linkedmdb.org/

reasoning section help us analyze how the users understand the result derivation process and if the explanation provided in the learning section had any impact on their understanding.

In the survey section of our study, we asked the participants if explanations help users to understand the result derivation and to make trust judgments on the results. Furthermore, we asked them which types of information they think are helpful in an explanation for understanding and making trust judgments. The questions in the survey section help us understand how the participants feel about the system and its explanation features.

5.3.2 Setup and Participants

The query we used is a query to find the British movies with American actors, shown in Listing 5.1. Part of the query is solved in LinkedMDB (lines 4–6: finding the British movies) and part of it is solved in DBpedia (line 7: finding birth places of the actors).

Listing 5.1: SPARQL query for finding British Movies with American Actors

```
1 SELECT  ?film ?actor
2 WHERE
3   {
4     ?film mdb-movie:country mdb-country:GB.
5     ?film mdb-movie:actor ?actor.
6     ?actor owl:sameAs ?sameActor.
7     ?sameActor dbpedia-owl:birthPlace dbpedia:United_States
8   }
```

A result tuple we provide to participants includes URIs for a film and an actor. We intentionally do not provide natural language descriptions in a result tuple. Instead we provide URIs from LinkedMDB – which are numeric resource URIs – for an actor and a film. This is to make sure that participants are not using their background knowledge about movies and actors in their answers. For the data

source selection and source triple selection questions, we provide small subsets of DBpedia triples (11 triples) and LinkedMDB triples (13 triples). We used Google Forms[3] for the questionnaires and Google App Engine[4] to randomize the selection of two cases – "with explanation" or "without explanation".

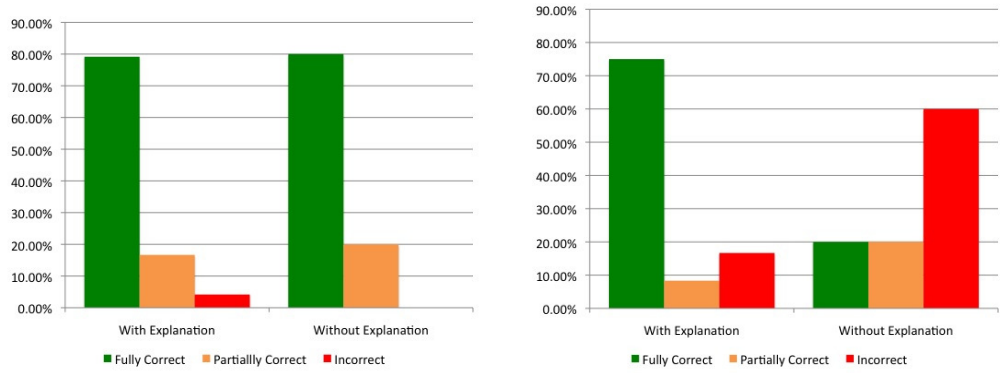
We invited the members of our laboratory[5] (via our mailing list), the members of Semantic Web Interest Group[6] (via their mailing list), and the followers of Twitter hashtags #SemanticWeb, #RDF, and #SPARQL. 11 participants took part in the study. There were 6 participants for the "with explanation" case and 5 participants for the "without explanation" case. There were 8 male participants and 3 female participants. The ages of the participants range from 22 to 65. All the participants had knowledge of RDF and SPARQL. The questionnaire and the responses of the participants are available online[7].

5.3.3 Results of the Study

We analyze the ability of the participants to apply their understanding of the system by computing the number of fully correct, partially correct, and incorrect answers for the data source selection and the source triple selection questions in the reasoning section. If a participant selects all the correct choices for an answer, we consider it as fully correct. If a participant selects all the correct choices but also selects some extraneous choices, we consider the answer as partially correct. If a participant's choices for an answer do not contain all the correct choices, we consider it as incorrect. In addition, if a participant selected all choices given for the source triple selection question, we consider the answer as incorrect to avoid guessing. For the data source selection question, we had 4 questions for 4 triple patterns in the query. We count the number of participants who provided fully correct answers, partially correct answers, and incorrect answers for each of these 4 questions. Then we take

[3] http://www.google.com/google-d-s/createforms.html
[4] https://appengine.google.com/
[5] http://wimmics.inria.fr/, https://glc.i3s.unice.fr/
[6] http://www.w3.org/2001/sw/interest/
[7] http://ns.inria.fr/ratio4ta/sqe/

the average of the counts for the fully correct answers, the average of the counts for the partially correct answers, and the average of the counts for the incorrect answers. These averages represent the average number of participants into the three answer categories categories – fully correct, partially correct, and incorrect – for the data source selection question as a whole. We compute these averages separately for both the “with explanation" and “without explanation" cases and compute the percentages of participants in the three answer categories for the two cases from these average.



(a) Data source selection

(b) Source triple selection

Figure 5.1: Participants’ response about data source selection and source triple selection.

5.3.3.1 Users’ Understanding and Trust

Figure 5.1(a) shows the percentage of participants with fully correct, partially correct, and incorrect answers when the explanation is given and when the explanation is not given for the data source selection question. The results are very similar for both “with explanation" and “without explanation" cases. 79.17% of participants provided fully correct answers when the explanation was given. 80.0% of participants provided fully correct answers when the explanation was not given. 20% of the participants provided partially correct answers and 4.17% provided incorrect answers when the explanation was given. 16.67% of participants provided partially correct answers and 0% provided incorrect answers when the explanation was not

given. The majority of the participants understood how data source selection works for our federated query processor system when the explanation was given (79.17%) and also when the explanation was not given (80.0%). Therefore the impact of explanations for source selection understanding is not clear from our study.

For the source triple selection question, we had two questions for the two data sources we used. We compute the percentages of participants in the fully correct, partially correct, and incorrect answer categories for the “with explanation" and “without explanation" cases using the same method as the data source selection question. Figure 5.1(b) shows the percentage of participants with fully correct, partially correct, and incorrect answers when the explanation is given and when the explanation is not given for the source triple selection question. More participants provided correct answers when the explanation was given (75% for “with explanation", 20% for “without explanation"). Furthermore, more participants provided incorrect answers when the explanation was not given (16.67% for “with explanation", 60% for “without explanation"). This clearly shows that participants who were given explanations understood better which triples contributed to the result from the two data sources.

The final question in the reasoning section asks participants to rate their confidence level about the answers for the data source selection question and the source triple selection question. Figure 5.2 shows the confidence level of the participants about their answers. 50.0% of participants with explanation rate their confidence as very high whereas none of participants without explanation rate very high. 33.33% of participants with explanation rate their confidence as high whereas 80% of participants without explanation rate high. This shows that participants with explanation are more confident in their answers – as many of them answered “very high" or “high".

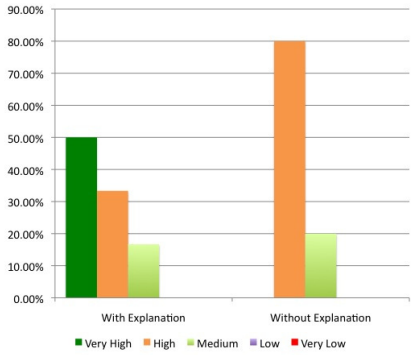
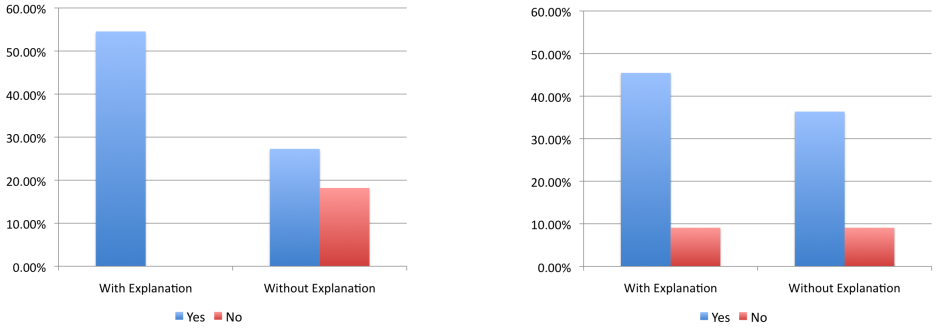


Figure 5.2: Participants' confidence level about their answers.



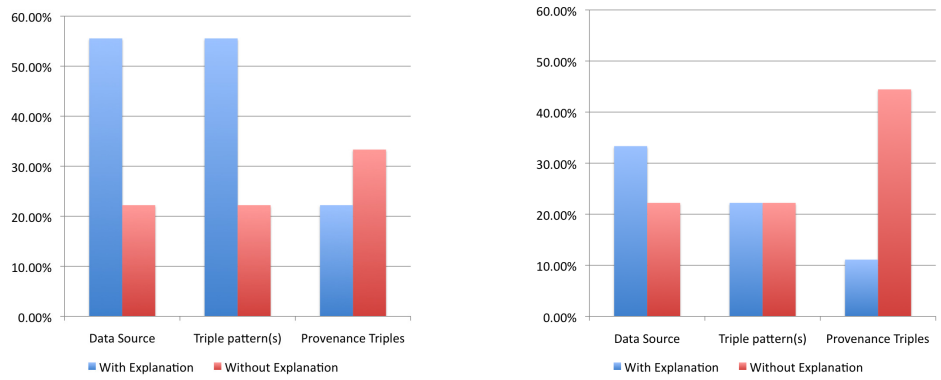
(a) Understanding

(b) Making trust judgments

Figure 5.3: Percentage of participants who answered that explanations are helpful ("yes") or unhelpful ("no").

5.3.3.2 How Users Feel About the System

For the survey section, we asked the participants if explanations are helpful to understand the query result derivation, and if explanations are helpful to make trust judgments on the query result. If a participant answered "yes", he/she was also asked what kind of information he/she found helpful. Figure 5.3(a) shows the percentage of participants who answered that explanations are helpful ("yes") or unhelpful ("no") for understanding the query result derivation. 54.55% of the participants who answered "yes" were provided with the explanation and 27.27% of them were not provided with the explanation. This clearly shows that there is a positive impact of explanations for understanding the query result derivation. Note that none of



(a) Understanding (b) Making trust judgments

Figure 5.4: Participants who found different types of information in the explanation helpful.

the participants answered explanations are unhelpful ("no") when they were provided with the explanation. 18.18% of the participants who answered "no" were not provided with the explanation. This means that the majority of the participants (yes: 81.82%, no:18.18%) – irrespective of whether they were provided with the explanation or not – feel are explanations helpful for understanding the query result derivation. Figure 5.3(b) shows the percentage of participants who answered if explanations are helpful make trust judgments on the query result. 45.455% of the participants who answered "yes" were provided with the explanation and 36.36% of them were not provided with the explanation. 9.09% of the participants who answered "no" were provided with the explanation and 9.09% of them were not provided with the explanation. Again, the majority of the participants (yes: 81.82%, no:18.18%) feel that explanations are helpful to make trust judgments on the query result irrespective of whether they were provided with the explanation or not. The 9.095% higher value for the cases of "with explanation" for the "yes" answer shows that there was indeed a positive impact of explanations for making trust judgments on the query result.

Figure 5.4(a) shows the participants who found the information about data sources, triple patterns, and *why-provenance* triples helpful for understanding the

query result derivation. Note that only the answers from participants who answered “yes" for the question shown in Figure 5.3(a) are considered. Out of 9 participants who answered “yes", a total of 77.78% responded that the data source related information was helpful, 55.56% were provided with the explanation and 22.22% were not provided with the explanation; 77.78% responded that the triple pattern related information was helpful, 55.56% were provided with the explanation and 22.22% were not provided with the explanation; and 55.55% responded that the provenance triple related information was helpful, 22.22% were provided with the explanation and 33.33% were not provided with the explanation. This shows that providing information about data sources and triple patterns had a positive impact for understanding. However, only 22.22% with explanation responded that providing information about provenance triples were helpful for understanding. Though, our analysis on source selection question responses (Figure 5.1(b)) shows that the explanation helped participants significantly improve their correctness on selecting the provenance triples. Therefore, it is hard to explain why only 22.22% with explanation responded that the provenance triple related information was helpful. One possible reason could be that when they were not given the explanation, they felt the need for explanations with provenance triple (hence 33.33% for without explanation). But when they were given the explanation, they were not aware that the provenance triple related information helped them to have a better understanding. Figure 5.4(b) shows the participants who found the information about data sources, triple patterns, and *why-provenance* triples helpful to make trust judgments. Again only the answers from participants who answered “yes" for the question shown in Figure 5.3(b) are considered. Out of 9 participants who answered “yes", a total of 55.55% responded that the data source related information was helpful, 33.33% were provided with the explanation and 22.22% were not provided with the explanation; 44.44% responded that the triple pattern related information was helpful, 22.22% were provided with the explanation and 22.22% were not provided with the expla-

nation; and 55.55% responded that the provenance triple related information was helpful, 11.11% were provided with the explanation and 44.44% were not provided with the explanation. This shows that the data source related information had a positive impact for making trust judgment. But the impact of triple pattern related information for making trust judgment is not clear. Again, it is interesting to notice that participants who were not given the explanation felt the need for provenance triples related information. This analysis in Figure 5.4 shows that the participants found data source and triple pattern related information helpful for understanding the query result derivation, but have less strong feeling about provenance triples related information for understanding query result derivations. For making trust judgments, participants do not have as strong opinions, but the majority of them feel that data source and provenance triple related information are helpful.

5.3.4 Discussion and Implications

Although the impact of explanations for data source selection was not clear from our study, percentage of correct answers for both cases is high and their difference is low (explanation: 79.17%, without explanation: 80.0%). Participants who were given explanations understood better which triples contributed to the result from the two data sources. This means that participants with explanation apply their understanding of the system they learned from the explanations. In other words, the participants who were given explanations understood the system better than the participants without explanation. The majority of the participants feel that explanations are helpful to understand query result derivations and to make trust judgments on query results. Also the participants with explanation were more confident on their answers. Therefore, we can say the explanations helped the participants to better understand the system and helped them make better trust judgments on the results. This validates our hypothesis (**H1**) that query result explanations improve user experience over having no explanations – where user experience is defined as

understanding and trust.

5.4 Summary

In this chapter, we presented a user study to evaluate the impact of query result explanations in a federated query processing scenario for Linked Data . Our user study shows that our query result explanations are helpful for end users to understand the result derivations and make trust judgments on the results.

In the next chapter, we present an approach to explain results produced by applications that consume Linked Data. The consumed data in this context can be also some derived data by other applications. Therefore we discuss explaining not only the reasoning by the applications but also the derivations of consumed data.

Chapter 6

Linked Explanations

Contents

In Chapter 4 and Chapter 5, we discussed explanations for query results. In this chapter, we discuss explanations for results produced by applications that consume Linked Data. The consumed data by Linked Data applications can be also some derived data by other applications. We discuss explaining not only the reasoning by the applications but also the derivations of consumed data. We discuss how publishing explanation metadata enables a decentralized approach to explanations for distributed reasoning. We introduce a vocabulary to describe explanation metadata and provide guidelines to publish explanation metadata as Linked Data.

6.1 Introduction

Applications can consume Linked Data, some of which can be derived by other applications, and reason on their consumed data to produce results or even produce more Linked Data. In this distributed scenario of Linked Data, it is essential to explain not only the reasoning by the applications but also the derivations of the consumed data, to help users – such as knowledge engineers or end-users of Linked Data applications – to understand how results or new Linked Data were derived. Much of the previous work on explanations for the Semantic Web does not address explanation in a distributed environment. The Inference Web [McGuinness 2003] approach proposes a centralized registry based solution for publishing explanation metadata from distributed reasoners. We propose a decentralized solution to this problem. In relation to the research questions in Section 1.2, we address the research question **RQ4**: "How to provide explanations for results produced by applications that consume Linked Data"?

To enable explanations for results produced by Linked Data data applications in a decentralized fashion, we publish explanation related metadata as Linked Data which we call Linked Explanations. In this approach, we are not constrained to publish the explanation metadata in a centralized location as in the Inference Web approach. To generate explanations, we retrieve the metatada by following their dereferenceable URIs and present them in a human understandable form. For publishing explanation related metadata, we present a vocabulary to describe explanation metadata and guidelines to publish these metadata as Linked Data. In contrast to explanations for SPARQL query result derivations that we discussed in Chapter 4 and Chapter 5, in this chapter we provide explanations for results produced by generic rule-based Linked Data applications. This means that we provide explanations for result derivations showing the triples used in a derivation. Furthermore, if those used triples were also derived, we provide explanations for them.

6.1.1 Publications

We published the results of this chapter in a full research paper [Hasan 2014a] and in a doctoral symposium paper [Hasan 2014b] at the Extended Semantic Web Conference 2014 (ESWC2014); and in a full research paper [Hasan 2012a] in the Semantic Web Collaborative Spaces Workshop 2012 (SWCS 2012) at the 21st International World Wide Web Conference 2012 (WWW 2012).

6.2 Explanation Approaches for the Semantic Web

As we discuss in Chapter 2, there have been a number of previous works on explaining reasoning in the Semantic Web literature. Table 6.1 shows a comparison of important previous works considering the criteria below:

Domain Independence. Indicates if a work is designed to support domain independent scenarios or application specific scenarios.

Linked Data Support. Indicates if a work supports explaining data published as Linked Data.

Distributed Reasoning. Indicates if a work supports explaining distributed reasoning. For example, chains of applications can use data which was derived by other applications distributed across the Web, and the produce new derived data and publish them. This criterion indicates if the work supports explanation in such scenarios.

Decentralization. Indicates if the explanation infrastructure is decentralized or centralized.

Standard Languages. Indicates if explanation metadata is represented using standard languages such as RDF or XML.

Inference Web [McGuinness 2003, McGuinness 2008, McGuinness 2004] proposes a centralized registry based solution for publishing explanation metadata

	Inference Web	WIQA	KOIOS	Justifications	OntoNova	KiWi
Domain Independence	√			√	√	√
Linked Data Support						
Distributed Reasoning	√					
Decentralization						
Standard Languages	√	√	√			

Table 6.1: Comparison of works on explanation for the Semantic Web.

from distributed reasoners. In fact, none of previous works support decentralized approach to explanation. In contrast, we propose a decentralized solution to address explanations in the distributed setting of Linked Data. Both WIQA [Bizer 2007] and KOIOS [Forcher 2010] provide application specific explanations which include process descriptions of specific algorithms. In contrast, our explanations are suitable for generic Linked Data scenarios. Justification related works [Horridge 2008, Horridge 2011, Kalyanpur 2007, Horridge 2009, Wang 2005, Kalyanpur 2005, Meyer 2006, Schlobach 2003, Lam 2008, Moodley 2010, Buneman 2010, Flouris 2009], OntoNova [Angele 2003] , and Knowledge in a Wiki (KiWi) [Kotowski 2010] do not represent their explanation metadata using standard data formats. This is an undesirable situation for Linked Data scenarios because data consumers would not be able to process such non standard explanation metadata. None of these previous works support explanation for Linked Data.

6.3 Explanations for Linked Data

To enable explanations for Linked Data, we publish the explanation metadata (along with the data) as Linked Data. We describe the explanation metadata using our

proposed vocabulary *Ratio4TA*[1]. We generate explanations by retrieving the explanation metatada by following their dereferenceable URIs and presenting them in a human understandable form.

6.3.1 Representing Explanation Metadata

Proof Markup Language (PML) [Pinheiro da Silva 2006] and the AIR Justification Ontology (AIRJ) [Kagal 2011] are important previous works on representing explanation metadata. PML allows describing provenance metadata, justifications for derivations of conclusions, and trust related metadata. Additionally, a light weight variant of PML known as PML-Lite [Pinheiro da Silva 2008] presents a simple subset of PML. AIRJ extends PML-Lite and provides primitives to represent the different events and the operations performed by reasoners. PML and AIRJ use RDF container concepts. RDF containers use blank nodes to connect a sequence of items [RDF 2014b]. However, as a common practice, blank nodes are avoided while publishing Linked Data [Heath 2011]. It is not possible to make statements about blank nodes as they do not have identifiers. Therefore, blank nodes make data integration harder in the global dataspace of Linked Data. Additionally, the existing ontologies do not use any common data interchanging standard such as W3C PROV-O. This makes it hard for applications across the Web to make sense of the explanation metadata. VoID [Alexander 2009] is vocabulary for describing metadata about RDF data sets. These metadata can include access metadata (metadata about methods to access the actual triples in a data set) and structural metadata (e.g. vocabularies used, statistics about the size of the data set). The *Dataset* concept is the core concept of VoID. It represents a RDF data set containing a set of triples. The Dataset concept is used to make statements about an entire RDF data set. In contrast to VoID, our goal is to associate explanation related metadata for data with different levels of granularity. To address these issues, we introduce a new vocabulary to describe explanation metadata next.

[1] http://ns.inria.fr/ratio4ta/

6.3.1.1 *Ratio4TA* Vocabulary

Ratio4TA (inter*l*inked explan*ati*ons *for* *t*riple *a*ssertions) is an OWL ontology for describing explanation metadata. *Ratio4TA* extends the W3C PROV Ontology[2]. This promotes interoperability by enabling data consumers to process explanation metadata according to W3C PROV standards. We use the named graph[3] mechanism [Carroll 2005] to make statements about RDF triples – the notion of named graphs is also adapted in the specification of RDF 1.1 [RDF 2014a]. Using named graphs allows us to associate explanation metadata for data with different levels of granularity – explanation metadata for a triple or a graph containing more than one triple. Furthermore, we use named graphs to group together explanation metadata and make the metadata for an explanation referenceable by a single URI. Applications can expose their explanation metadata using *Ratio4TA* to enable other applications to consume machine processable explanations. Consumers of the explanation metadata can use their preferred tools to present and visualize explanations. Figure 6.1 shows the core concepts and relations of *Ratio4TA*. They allow describing data, reasoning processes, results, data derivations, rules, and software applications. *Ratio4TA* includes the following core classes:

Data: A *Data* is a set of RDF statements. The *Data* class is a sub-class of the *prov:Entity* class and the *rdfg:Graph*.

InputData: An *InputData* represents an input data (a set of RDF statements) used by a reasoning process. *InputData* is a sub-class of *Data*.

OutputData: An *OutputData* represents an output data (a set of RDF statements) by a reasoning process. *OutputData* is a sub-class of *Data*.

ReasoningProcess: A *ReasoningProcess* represents a reasoning process of a soft-

[2]W3C PROV Ontology: `http://www.w3.org/TR/prov-o/`. We use the prefix *prov* for the classes and the properties of PROV.

[3]We use the prefix *rdfg* for the classes and the properties of the named graph vocabulary (`http://www.w3.org/2004/03/trix/rdfg-1/`).

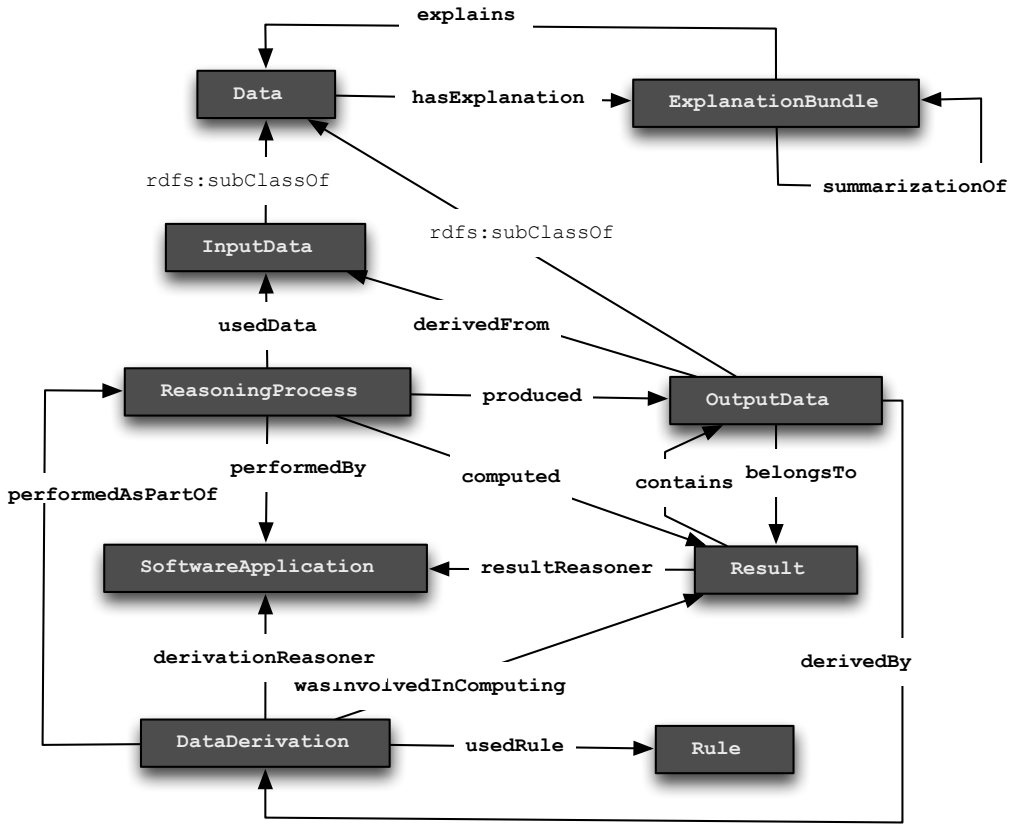


Figure 6.1: The core classes and properties of *Ratio4TA*.

ware application. A reasoning process uses *InputData* and computes results. Each of these computed results includes *OutputData*. More specifically, a *ReasoningProcess* produces *OutputData*. *ReasoningProcess* is a sub-class of *prov:Activity*.

Result: A *Result* represents a result computed by a reasoning process. A *Result* contains *OutputData*. An *OutputData* belongs to a *Result*. *Result* is a sub-class of *prov:Association* and *prov:Generation*.

DataDerivation: A *DataDerivation* represents a data derivation that is performed as part of a *ReasoningProcess*. *DataDerivation* may use a rule and may be involved in computing a *Result*. *DataDerivation* is a sub-class of *prov:Derivation* and *prov:Association*.

SoftwareApplication: A *SoftwareApplication* consumes and produces data. A *SoftwareApplication* can perform reasoning processes. A *ReasoningProcess* can have data derivations as it's parts. Therefore, the reasoner for a *DataDerivation* is a *SoftwareApplication*. *SoftwareApplication* is a sub-class of *prov:SoftwareAgent*.

Rule: A *Rule* represents a rule that a *ReasoningProcess* uses for a *DataDerivation*. *Rule* is a sub-class of *prov:Plan*. The encoding of rules is out of the scope of our work. However, for rules implemented using SPARQL, our proposal is to use SPIN[4] for representing them in RDF.

ExplanationBundle: An *ExplanationBundle* is a set of RDF statements which represent the explanation metadata for a *Data*. *ExplanationBundle* is a sub-class of *rdfg:Graph* and *prov:Bundle*.

Figure 6.2 shows the relationships between the classes of *Ratio4TA* and the classes of PROV. The white boxes show the classes of PROV and the black boxes

[4]http://spinrdf.org/

show the classes of *Ratio4TA*. All the classes of *Ratio4TA* are defined as sub-classes of the classes of PROV.

The properties of *Ratio4TA* are defined as sub-properties of the properties of PROV ontology. Table 6.2 shows the descriptions of the properties of *Ratio4TA*.

Table 6.2: Properties of *Ratio4TA*.

Property	**Description**
belongsTo	A reasoning process uses input data and computes results. Each of these computed results includes output data. The *belongsTo* property specifies that an output data belongs to a result. *belongsTo* is a sub-property of *prov:qualifiedGeneration*. The domain of *belongsTo* is *OutputData* and the range of *belongsTo* is *Result*. *belongsTo* is defined as the inverse property of the *contains* property.
computed	A reasoning process computes results. The *computed* property specifies that a reasoning process computes a result. *computed* is a sub-property of *prov:qualifiedAssociation*. The domain of *computed* is *ReasoningProcess* and the range of *computed* is *Result*.
contains	A reasoning process uses input data and computes results. Each of these computed results includes output data. The *contains* property specifies that a result contains an output data. *contains* is defined as the inverse property of the *belongsTo* property.

Table 6.2: Properties of *Ratio4TA*.

Property	**Description**
derivationReasoner	A software application performs derivations. The *derivationReasoner* property specifies that a data derivation is performed by a software application. *derivationReasoner* is a sub-property of *prov:agent*. The domain of *derivationReasoner* is *DataDerivation* and the range of *derivationReasoner* is *SoftwareApplication*.
derivedBy	A data derivation uses rules and derives output data. The *derivedBy* property specifies that an output data is derived by a data derivation. *derivedBy* is a sub-property of *prov:qualifiedDerivation*. The domain of *derivedBy* is *OutputData* and the range of *derivedBy* is *DataDerivation*.
derivedFrom	A data derivation transforms a data into another, constructs a data into another, or updates a data, resulting in a new one. Note that by data we mean an instance of the *Data* class. The *derivedFrom* property specifies that a data is derived from a data. *derivedFrom* is a sub-property of *prov:wasDerivedFrom*. The domain of *derivedFrom* is *Data* and the range of *derivedFrom* is *Data*.
hasExplanation	An explanation bundle contains statements about how an instance of data was derived. The *hasExplanation* property specifies that a data is explained by an explanation bundle. *hasExplanation* is a sub-property of *prov:has_provenance*. The domain of *hasExplanation* is *Data* and the range of *hasExplanation* is *ExplanationBundle*. *hasExplanation* is defined as the inverse property of the *explains* property.

Table 6.2: Properties of *Ratio4TA*.

Property	**Description**
explains	An explanation bundle contains statements about how an instance of data was derived. The *explains* property specifies that an explanation bundle explains a data. *explains* is defined as the inverse property of the *hasExplanation* property.
performedAsPartOf	A reasoning process performs data derivations. The *performedAsPartOf* specifies that a data derivation is performed as part of a reasoning process. *performedAsPartOf* is a sub-property of *prov:hadActivity*. The domain of *performedAsPartOf* is *DataDerivation* and the range of *performedAsPartOf* is *ReasoningProcess*.
performedBy	A software application performs a reasoning process. The *performedBy* property specifies that a reasoning process is performed by a software application. *performedBy* is a sub-property of *prov:wasAssociatedWith*. The domain of *performedBy* is *ReasoningProcess* and the range of *performedBy* is *SoftwareApplication*.
produced	A reasoning process computes results and a computed result contains output data. The *produced* property specifies that a reasoning process produced an instance of output data. *produced* is a sub-property of *prov:generated*. The domain of *produced* is *ReasoningProcess* and the range of *produced* is *OutputData*.

Table 6.2: Properties of *Ratio4TA*.

Property	**Description**
resultReasoner	*resultReasoner* is a sub-property of *prov:agent*. The domain of *resultReasoner* is *Result* and the range of *resultReasoner* is *SoftwareApplication*.
summarizationOf	A summarized explanation bundle can contain the most important information from an explanation bundle. The *summarizationOf* property specifies an explanation bundle is a summarization of a explanation bundle. *summarizationOf* is a sub-property of *prov:generalizationOf*. The domain of *summarizationOf* is *ExplanationBundle* and the range of *summarizationOf* is *ExplanationBundle*.
usedData	A reasoning process uses input data to compute its results. The *usedData* property specifies a reasoning process used an instance of input data. *usedData* is a sub-property of *prov:used*. The domain of *usedData* is *ReasoningProcess* and the range of *usedData* is *InputData*.
usedRule	Data derivations use rules to perform derivations. The *usedRule* property specifies that a data derivation used a rule. *usedRule* is a sub-property of *prov:hadPlan*. The domain of *usedRule* is *DataDerivation* and the range of *usedRule* is *Rule*.

Table 6.2: Properties of *Ratio4TA*.

Property	**Description**
wasInvolvedComputing	A reasoning process performs data derivations to compute results. The *wasInvolvedComputing* property specifies that a data derivation was involved in computing a result. *wasInvolvedComputing* is a sub-property of *prov:hadGeneration*. The domain of *wasInvolvedComputing* is *DataDerivation* and the range of *wasInvolvedComputing* is *Result*.

Figure 6.3 shows the relationships between the properties of *Ratio4TA* and the properties of PROV. All the properties except *explains* and *contains* and defined as direct sub-properties of PROV properties. *explains* is defined as an inverse property of *hasExplanation* and *contains* is defined as an inverse property of *belongsTo*.

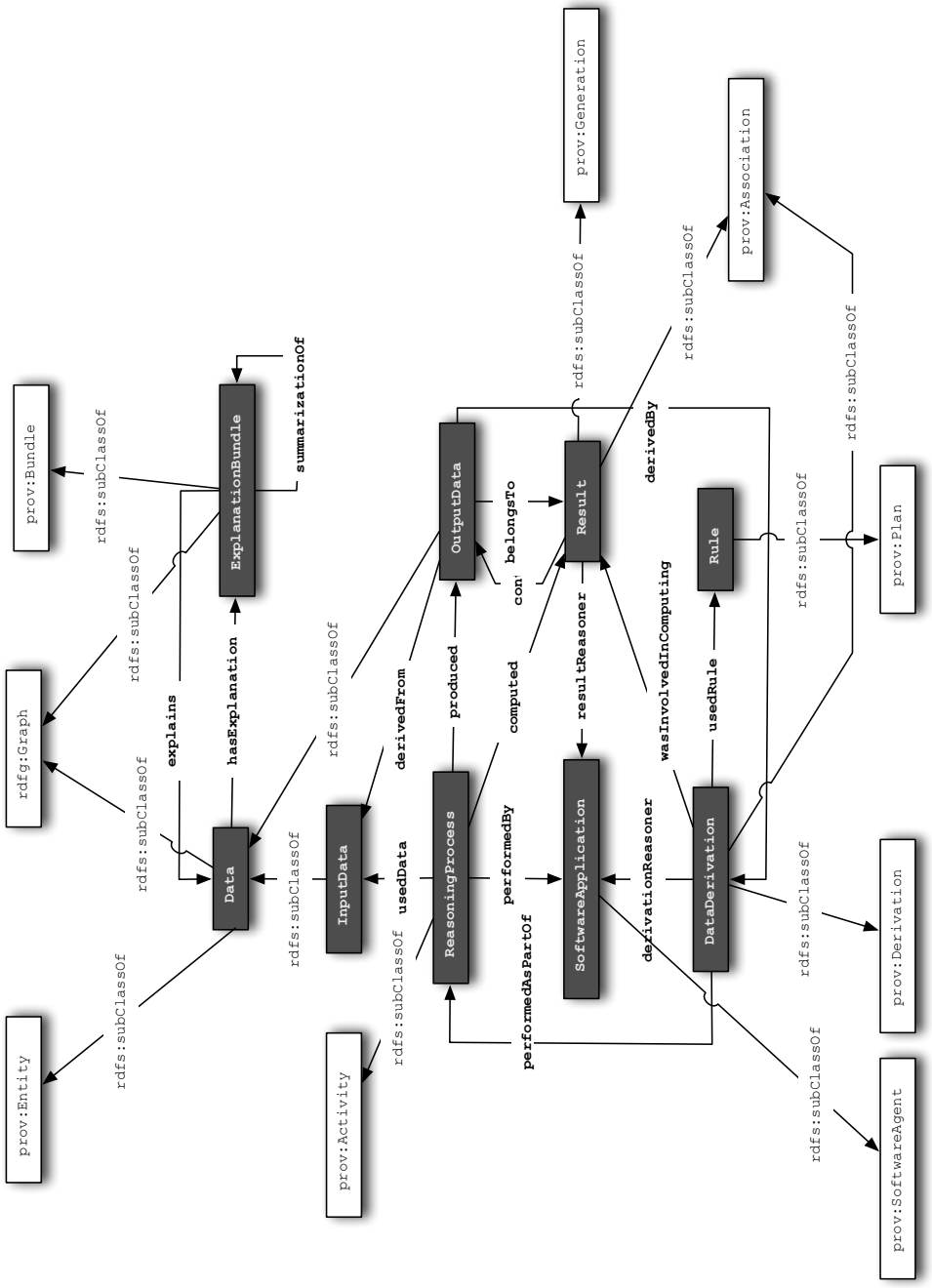


Figure 6.2: *Ratio4TA* and its relationships with the classes of PROV.

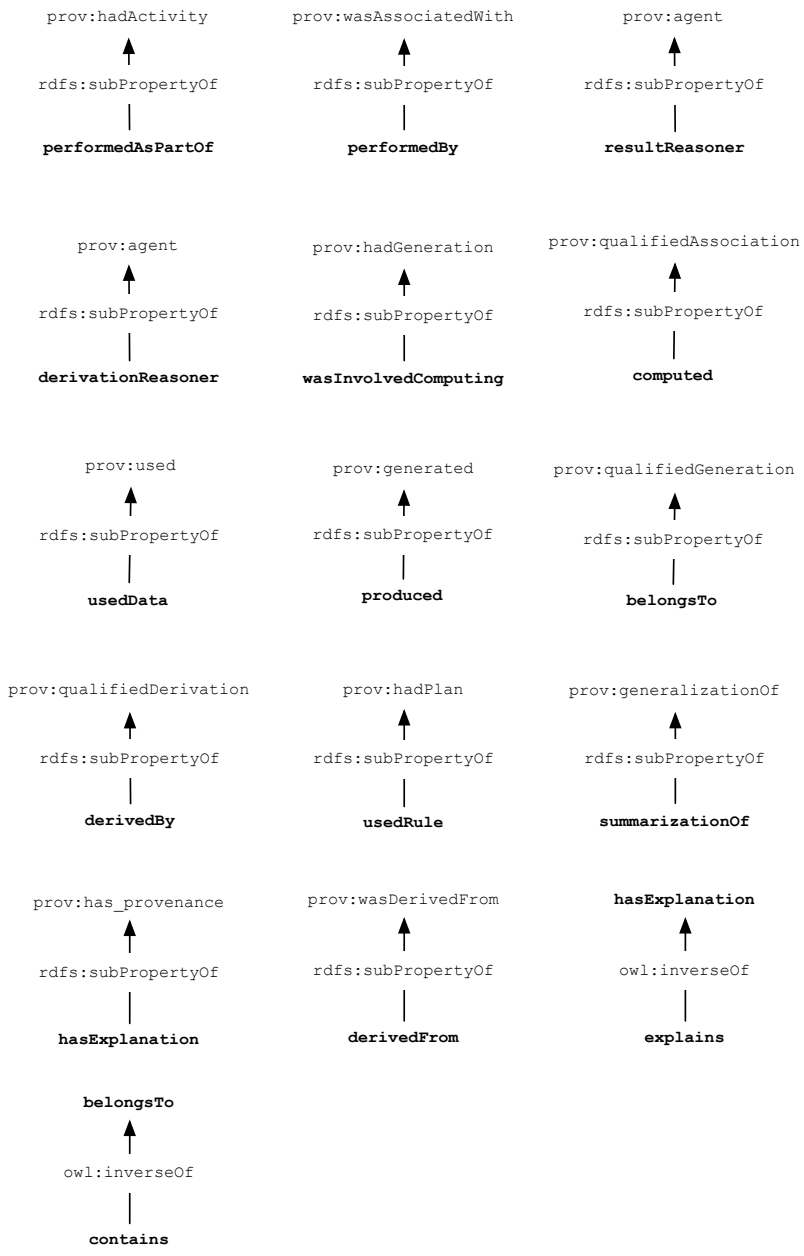


Figure 6.3: The properties of *Ratio4TA* and their relationships with the properties of PROV.

We provide the source code of *Ratio4TA* in Appendix A. The source code can be also downloaded from the online specification document of *Ratio4TA* located at http://ns.inria.fr/ratio4ta/.

6.3.1.2 Example of Encoding using *Ratio4TA*

Listing 6.1 shows an extract of an explanation described using *Ratio4TA* in TriG [Bizer 2014, Carroll 2005] notation. The example in this listing shows the explanation metadata for the derived triple *lodapp:data1*. The named graph *lodapp:explanation1* contains the explanation metadata. The metadata include links to the reasoning process, the input data, the rule, the software application, and the result to which the derivation contributes. Lines 29–31 show the named graph *lodapp:data1* which contains the derived triple (line 30). Lines 2–27 show the named graph *lodapp:explanation1* which contains the explanation metadata for the derivation. Line 3 specifies that *lodapp:explanation1* explains *lodapp:data1*. Lines 5–8 in *lodapp:explanation1* show the related type declarations – we do not show all the type declarations in this example for the purpose of simplification. Lines 10–14 show the encoding of the reasoning process *lodapp:reasoningProcess1* that produced *lodapp:data1*. Line 10 specifies that the reasoning process *lodapp:reasoningProcess1* was performed by the software application *lodapp:corese*. Lines 11–12 specify that the reasoning process *lodapp:reasoningProcess1* used *lodapp:inputData1* and *lodapp:inputData2*. Line 13 specifies that the *lodapp:reasoningProcess1* computed the result *lodapp:result1*. Line 14 specifies that the *lodapp:reasoningProcess1* produced the data *lodapp:data1*. The encodings of *lodapp:inputData1* and *lodapp:inputData2* are shown in lines 33–35 and lines 37–39 respectively. Line 16 specifies that the reasoner for the result *lodapp:result1* is *lodapp:corese*. Lines 18–19 specify that the data *lodapp:data1* was derived from *lodapp:inputData1* and *lodapp:inputData2*. Line 20 specifies that the data *lodapp:data1* belongs to the result *lodapp:result1*. Line 21 specifies that the data *lodapp:data1* was derived by the derivation *lodapp:derivation1*. Lines 23–27 show the encoding of the derivation *lodapp:derivation1*. Line 23 specifies that the derivation *lodapp:derivation1* used the rule *lodapp:geoFeatureRule*. Line 24 specifies that the derivation *lodapp:derivation1* was involved in computing the result *lodapp:result1*. Line 25 specifies that the reasoner for the derivation

lodapp:derivation1 is the software application *lodapp:corese*. Line 26 specifies that the derivation *lodapp:derivation1* was performed as a part of the reasoning process *lodapp:reasoningProcess1*.

Listing 6.1: Extract from the explanation metadata for a derivation

```
1  # Explanation Metadata
2  lodapp:explanation1 {
3    lodapp:data1 r4ta:hasExplanation lodapp:explanation1.
4    # Type declarations
5    lodapp:explanation1 rdf:type r4ta:ExplanationBundle.
6    lodapp:corese rdf:type r4ta:SoftwareApplication.
7    ....
8    ....
9    # Reasoning process
10   lodapp:reasoningProcess1 r4ta:performedBy lodapp:corese;
11       r4ta:usedData lodapp:inputData1;
12       r4ta:usedData lodapp:inputData2;
13       r4ta:computed lodapp:result1;
14       r4ta:produced lodapp:data1.
15   # Computed result
16   lodapp:result1 r4ta:resultReasoner lodapp:corese .
17   # Output data
18   lodapp:data1 r4ta:derivedFrom lodapp:inputData1;
19       r4ta:derivedFrom lodapp:inputData2;
20       r4ta:belongsTo lodapp:result1;
21       r4ta:derivedBy lodapp:derivation1.
22   # Data derivation
23   lodapp:derivation1 r4ta:usedRule lodapp:geoFeatureRule;
24       r4ta:wasInvolvedInComputing lodapp:result1;
25       r4ta:derivationReasoner lodapp:corese;
26       r4ta:performedAsPartOf lodapp:reasoningProcess1.
27 }
28 # Derived data
29 lodapp:data1 {
30   dbpedia:Philadelphia gn:parentFeature geonames:5205788.
```

```
31 }
32 # Dbpedia data
33 lodapp:inputData1 {
34   dbpedia:Philadelphia owl:sameAs geonames:4560349 .
35 }
36 # GeoNames data
37 lodapp:inputData2 {
38   geonames:4560349 gn:parentFeature geonames:5205788.
39 }
```

Figure 6.4 shows the visualization of the example shown in Listing 6.1. The rectangles with dashed lines represent named graphs, the oval shapes represent resources, and the arrows represent properties. We omit the type declarations for the purpose of simplicity. As the figure shows, the reasoning process *lodapp:reasoningProcess1* is modeled inside the named graph *lodapp:explanation1*, which is an instance of the ExplanationBundle class, specifying its relationships with the software application *lodapp:corese*, the derivation *lodapp:derivation1*, the used rule *lodapp:geoFeatureRule*, the used input data *lodapp:inputData1* and *lodapp:inputData2*, the computed result *lodapp:result1*, and the produced output data *lodapp:data1*.

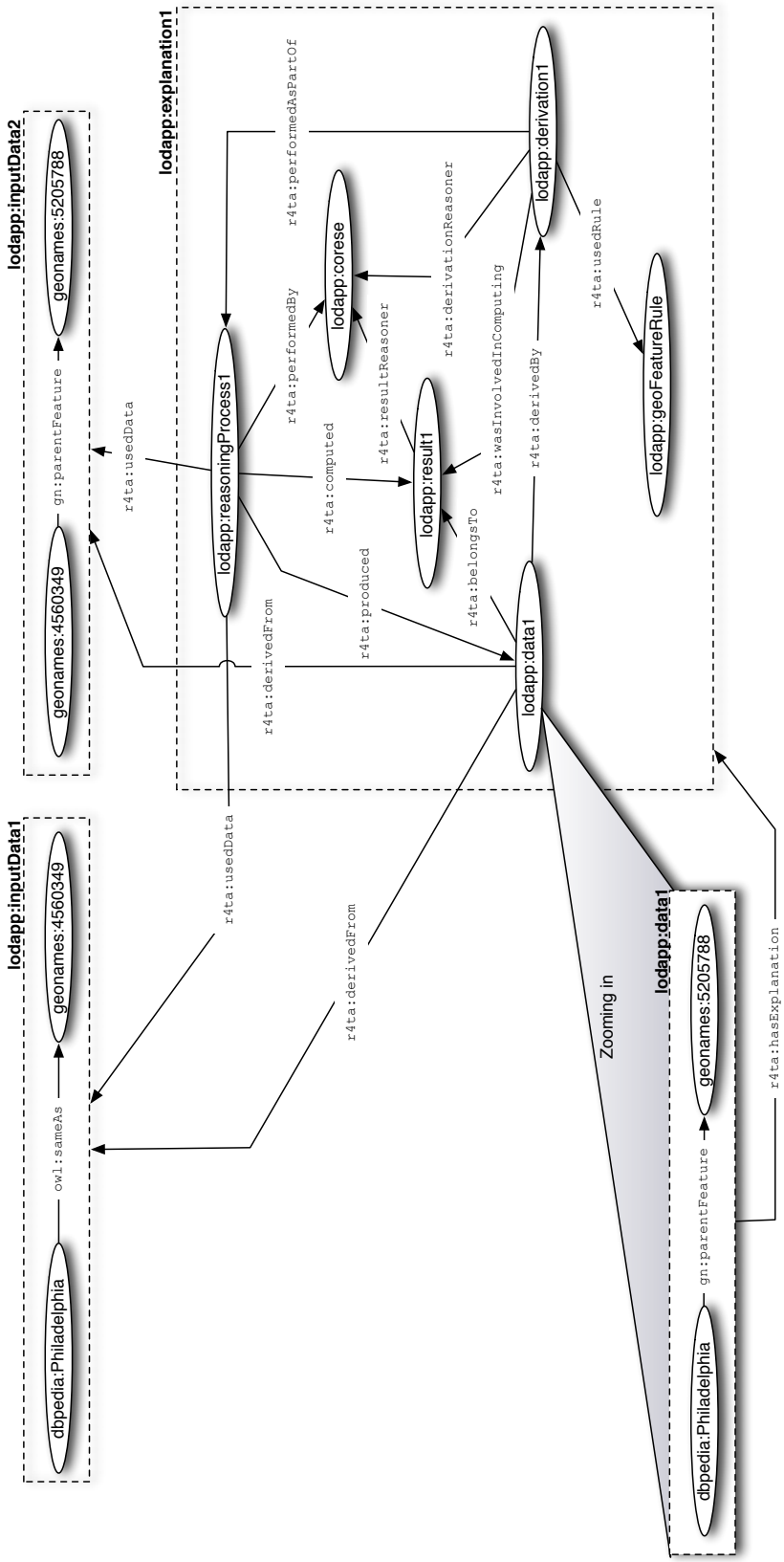


Figure 6.4: Visualization of the example shown in Listing 6.1.

6.3.2 Publishing Explanation Metadata: Linked Explanations

We publish the explanation metadata as Linked Data. This means that all the resources in our explanation metadata have dereferenceable HTTP URIs. This relates to the first and the second Linked Data principles (presented in Section 2.3.3). We avoid using blank nodes to keep the resources globally dereferenceable. It is important to note that our approach is dependent on named graphs for reification. The data triple(s) must be reifiable to specify explanation metadata for them. Also we group together triples for an explanation in a named graph. This ensures referencing to the metadata for an explanation using a single URI. When a URI for a named graph is dereferenced, we return all the triples inside that named graph, and all the triple that have the named graph URI as subject and as object. This ensures that we return the content of the named graph and the related contents of the named graph URI. When a URI for an RDF resource, that is not a named graph, is dereferenced, we return the triples that have the URI as subject and as object. This relates to the third and the fourth Linked Data principles, as we link related URIs (e.g. data is linked to explanations, explanations are linked to input data) and return them when some looks up a URI.

6.3.2.1 Principles for Linked Explanations

Considering above mentioned issues, we outline four principles for Linked Explanations, which are analogous to the Linked Data principles.

1. Use URIs as names of things, reified statements, and named graphs (RDF resources, reified data triples, and explanation metadata named graphs).

2. Use HTTP URIs, so that people can look up those names.

3. When a URI for a named graph (or a reified statement) is dereferenced, provide the statements inside that named graph, and all the statements that have the named graph URI as subject and as object. When a URI for an RDF resource,

that is not a named graph, is dereferenced, provide statements that have the URI as subject and as object.

4. Include links to other URIs (e.g. linking input and output data statements, and explanations metadata).

Using the Linked Explanations approach ensures that applications that are distributed across the Web can publish explanation metadata for their derived data. In addition, explanation metadata can be hosted anywhere in the Web and retrieved by URIs. Linked Data applications can consume data published using this approach with their explanation metadata to generate explanations. In essence, publishing explanation metadata following this approach enables a decentralized approach to explanations for distributed reasoning.

6.3.3 Accessing and Presenting Linked Explanations

We generate explanations from the published explanation metadata by recursively following the links between the involved explanation metadata and the data they describe. For a derived RDF statement *dst*, we crawl through the related metadata (by dereferencing their URIs) with a maximum depth limit and collect the set of explanation meta statements, and the set of RDF statements from which the derived RDF statement *dst* is derived. Our explanations are based on the notion of *proof tree* [Ferrand 2006]. *Proof trees* are abstract notions which are used in various domains in logic and computer science. Figure 6.5 shows an example of a *proof tree*.

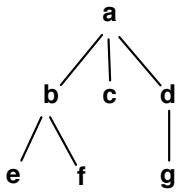


Figure 6.5: Example of a *proof tree*.

The *proof tree* in this example shows that **a** was directly derived from **b**, **c**, and **d**; **b** was directly derived from **e** and **f**; and **d** was directly derived from **g**. As **c**, **e**, **f**, and **g** are not derived from others, they are direct assertions. From an intuitive point of view {**b**, **c**, **d**} is an immediate explanation of **a**. The whole tree is a full explanation of **a**. In our proof tree-based explanations, each RDF statement is a node in the *proof tree*. A tree is *well founded* if it has no infinite branch. We use the maximum depth limit in our crawling process to keep our explanation *proof trees well founded*.

In the remainder of this thesis, we refer to the derived RDF statement (the initial *dst*) that we are explaining as the root statement rs. We refer to the set of all the RDF statements from which rs is derived (all the statements in the proof tree for rs) as knowledge statements KST. The RDF knowledge graph KG is the graph formed by union of KST and the root statement: $KG = RDFGraph(KST \cup rs)$. We generate natural language descriptions from the RDF statements in KG (using *rdfs:label* property values) and present them as explanations for human end-users.

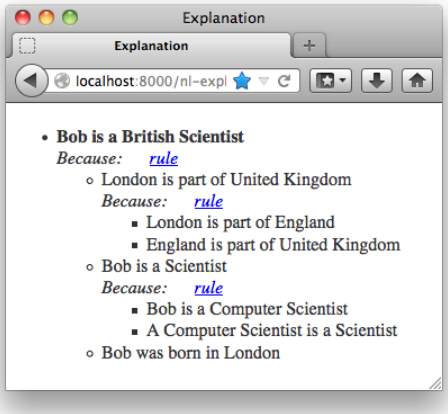


Figure 6.6: Example of a generated explanation.

Figure 6.6 shows an example of a generated explanation for a derived statement that "Bob is a British Scientist". Each derivation contains a link to the natural language representation of the used rule. As we mentioned before, the encoding

of rules are out of the scope of our work. However, our proposal is to encode rules in RDF and publishing them as Linked Data. This will allow to write the rules once, then enforcing them for derivations; linking them from the explanation metadata as they are also RDF resources with identifiers; and finally providing human understandable abstraction of them for explanation.

We illustrate the distributed and decentralized aspects of our approach using the same derivation "Bob is a British Scientist" in the scenario shown in Figure 6.7. We omit the namesapce prefixes in the figure for simplicity. *Data Source 1* publishes Linked Data about geographical locations. It contains two directly asserted triples: `:England :isPartOf :UnitedKingdom` and `:London :isPartOf :England`. It also contains the derived triple `:London :isPartOf :UnitedKingdom` (shown by the dashed arrow), which is derived from the other two triples in this data source. *Data Source 2* publishes Linked Data about people. It contains 3 directly asserted triples and 1 derived triple. The derived triple `:Bob rdf:type :Scientist` (shown by the dashed arrow) is derived from the triples `:Bob rdf:type :ComputerScientist` and `:ComputerScientist rdfs:subClassOf :Scientist` in this data source. The *Linked Data Application* consumes data from *Data Source 1* and *Data Source 2* and derives 3 new triples (shown by green and red dashed arrows). The derived triple `:Bob :birthPlace :England` is derived from `:Bob :birthPlace :London` and `:London :isPartOf :England`; same way the derived triple `:Bob :birthPlace :UnitedKingdom` is derived from `:Bob :birthPlace :London` and `:London :isPartOf :UnitedKingdom` (originally a derived triple in *Data Source 1*). The application produces `:Bob rdf:type :BritishScientist` as the result triple, which is derived from `:Bob :birthPlace :UnitedKingdom` (originally a derived triple in *Linked Data Application*) and `:Bob rdf:type :Scientist` (originally a derived triple in *Data Source 2*). This example shows how a Linked Data application can consume distributed data, which can be derived data, and derive its results.

Figure 6.8 shows how we can explain the result triple of the *Linked Data Application* using the Linked Explanations approach. The data sources can add the explanation metadata of their derived triples by following the Linked Explanations principles. This allows the*Linked Data Application* to follow the available `r4ta:hasExplanation` links of their consumed triples and discover the explanation metadata of those consumed triples.

Figure 6.7: Example of a Linked Data application.

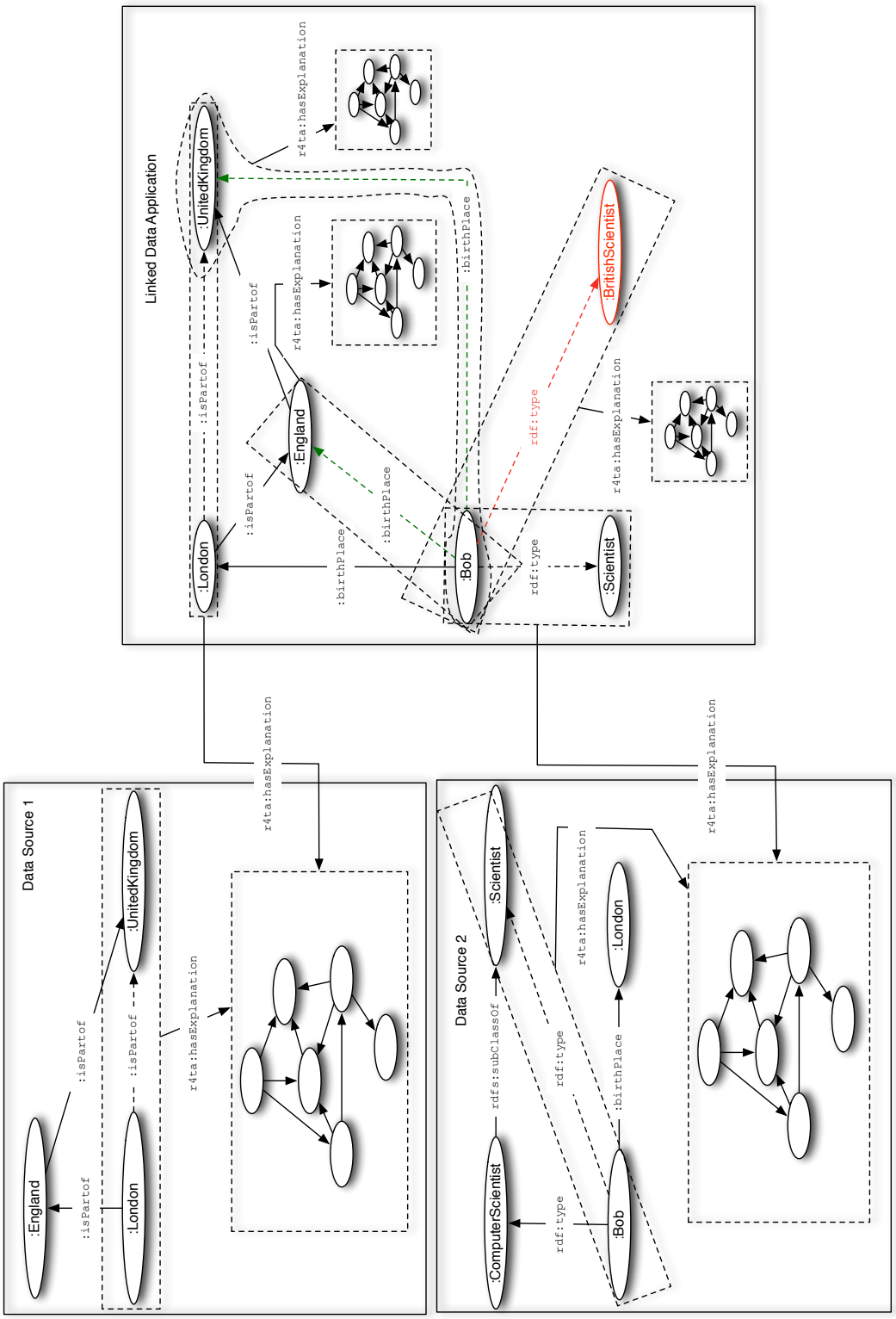


Figure 6.8: Example of Linked Explanations.

The *Linked Data Application* can also publish its result triple and explanation metadata by following the Linked Explanations principles. This scenario shows an example of explaining distributed reasoning. Furthermore, the explanation metadata is not published in centralized location. Each data source publishes its own explanation metadata – hence enables decentralization of explanation metadata.

6.4 Summary

In this chapter we discuss how to provide explanations for results produced by applications that consume Linked Data. We present the *Ratio4TA* vocabulary to describe explanation metadata. We introduce the notion of Linked Explanations and discuss how it enables explanations in distribute scenarios in a decentralized fashion. Finally, we discuss how to present Linked Explanations as proof tree-based explanations.

The proof tree-based full explanations generated from Linked Explanations can become very large which can overwhelm users. In the next chapter, we present a summarization approach for Linked Explanations.

Chapter 7

Summarizing Linked Explanations

Contents

In the previous chapter, we discussed how to provide proof tree-based full explanations for results produced by applications that consume Linked Data. These explanations generated from Linked Explanations can become very large which can overwhelm users with too much information. Users may need the possibility to transform long explanations into more understandable short explanations. Users may want to filter information in an explanation and focus on some specific kind of information in an explanation. In this chapter we present an approach to sum-

marize explanations and filter information in an explanation based on user specified explanation filtering criteria.

7.1 Introduction

Although explanations with the details of all the derivation steps may be useful for expert users, they may overwhelm non-expert users with too much information [Angele 2003, McGuinness 2004]. In addition, an expert user such as a knowledge engineer may want to focus on a specific part of a detailed explanation. A knowledge engineer may also want a short explanation to have an overview of the reasoning. We provide summarized explanations to address these problems. In relation to the research questions in Section 1.2, we address the research question **RQ5**: "How to summarize explanations for results produced by applications that consume Linked Data"? We define five summarization measures: (i) salience of RDF statements, (ii) similarity of RDF statements with respect to users' filtering criteria, (iii) abstractness of RDF statements with respect to the proof tree, (iv) subtree weight in the proof tree - weight of a node in the proof tree, (v) coherence of RDF statements with respect to the proof tree.

Recall that we generate explanations from the explanation metadata, published as Linked Explanations, by recursively following the links between the involved explanation metadata and the data they describe. For a RDF statement *dst*, we crawl through the related metadata with a maximum depth limit and collect the set of explanation meta statements, and the set of RDF statements from which the derived RDF statement *dst* is derived, which we refer to as the root statement rs. We refer to the set of RDF statements from which rs is derived as knowledge statements KST. The RDF knowledge graph KG is the graph formed by union of KST and the root statement: $KG = RDFGraph(KST \cup rs)$. Figure 7.1(a) shows an example of an explanation for a derived statement "Bob is a British Scientist".

Figure 7.1(b) shows an example of a summarized explanation for "Bob is a British

(a) Full explanation. (b) Summarized explanation.

Figure 7.1: Examples of a full explanation and a summarized explanation.

Scientist". Users can switch to the full explanation by clicking on the "more details" link. In this chapter, we discuss how we provide such summarized explanations.

7.1.1 Publications

We published the results of this chapter in a full research paper [Hasan 2014a] and in a doctoral symposium paper [Hasan 2014b] at the Extended Semantic Web Conference 2014 (ESWC2014).

7.2 Explanation and Summarization

Only Inference Web [McGuinness 2003, McGuinness 2004, McGuinness 2008] provides a summarization feature in their explanations. Inference Web allows zooming in for more details in the graphical explanation proof trees and zooming out for less details. But researchers have studied ontology summarization. RDF Sentence graph based summarization [Zhang 2007] extracts RDF sentences based on centrality measures. Our work has a similar approach to sentence graph summarization approach in the sense that we rank RDF statements based on some measures. However, we define new measures for summarizing explanations. Peroni *et al.* [Peroni 2008] discuss how to identify key concepts in an ontology. They draw summarization criteria from cognitive science (natural categories), network topology (density and coverage), and

lexical statistics (term popularity). Alani *et al.* [Alani 2006] discuss shrinking an ontology by analyzing the usage of the ontology. Alani *et al.* analyze the query log against an ontology to understand the important parts of the ontology. Peroni *et al.* and Alani *et al.* focus on a concept level summarization of ontologies. In contrast, our focus is on statement level summarization.

In [Angele 2003, McGuinness 2004], researchers discuss the importance of providing short explanations rather than overwhelming the end-users with too much information. The authors of [Angele 2003] also discuss filtering information in explanations in order to provide more relevant explanations.

7.3 Summarizing Explanations

We propose an approach to summarizing explanations taking into account user specified filtering criteria. More formally, let $KG = (R, T)$ be an RDF knowledge graph, where R is the set of resources and literals and T is the set of RDF statements. Let rs be the root statement (therefore the knowledge statements $KST = T \setminus rs$). We provide summarized explanations by summarizing RDF statements from KST. We use the term "oriented graph" to refer to KG throughout this chapter. Our summarization approach includes first a ranking step and then a re-ranking step. We rank the statements in an explanation based on their salience, similarity with respect to the user specified filtering criteria, and abstractness with respect to the proof tree. Then we refine this ranking by re-ranking the statements based on their subtree weight in the proof tree - weight of a node in the proof tree, and their coherence with respect to the proof tree. It is important to note that our summarized explanations may not always conform to the correctness of deductions from a logical point of view. Our summarized explanations are not aimed at explaining the correct deduction steps. Rather the aim is to provide a short overview of the background information used in a deduction. We describe below the measures we use for summarizing explanations.

7.3.1 Measures for Ranking

We rank the statements in KST based on their scores we compute using our summarization measures. The scores are normalized and range from 0.0 to 1.0. A higher score for a statement means that the statement is more suitable for a summary. Taking n statements, where $n <\mid KST \mid$, with scores greater than a threshold value gives a summarized list of statements which can explain rs. For the ranking step, we compute the scores by using three measures: salience (S_{SL}), similarity (S_{SM}), and abstractness (S_{AB})

7.3.1.1 Salient RDF Statements

The salience of an RDF statement indicates the importance of the RDF statements in the oriented graph. We use normalized degree centrality, $C_{DN}(v)$, to compute salience of RDF statements. Degree centrality of a vertex in a graph is the number of links the vertex has. We compute the salience $S_{SL}(i)$ of an RDF statement i using (7.1).

$$S_{SL}(i) = \theta_1 \times C_{DN}(subjectOf(i)) + \theta_2 \times C_{DN}(objectOf(i)) \tag{7.1}$$

In (7.1), $\sum_i \theta_i = 1$ and $\forall_i : \theta_i \geq 0$ *i.e.* we take the weighted average of the normalized degree centrality of the subject and the object of the RDF statement i. The $subjectOf(i)$ and the $objectOf(i)$ functions return respectively the subject resource and the object resource of the RDF statement i. We did not use the centrality of the predicate of statement while computing S_{SL} because we wanted an importance score representing the importance of the information in a statement but not the importance of the relation between the information. The centrality values of predicates in a RDF graph often do not change as they are directly used from the schemata. In contrast, every new RDF statement changes the centrality values of its subject and object.

7.3.1.2 Similar RDF Statements

The consumers of our explanations can specify a set of classes, FL, as their filtering criteria, where $FL \subseteq SC$ and SC is the set of all classes in the schemata used to describe KG. We rank the more similar statements to the concepts given in filtering criteria higher. We use the approximate query solving feature [Corby 2006] of Corese[1] to compute similarity. The approximate query solving feature is a semantic distance-based similarity feature to compute conceptual similarity between two classes in a schema. For a statement i and a set of classes as filtering criteria FL, we compute similarity $S_{SM}(i, FL)$ using (7.2).

$$\begin{aligned} S_{SM}(i, FL) = \theta_1 \times similarity_{node}(subjectOf(i), FL) \\ + \theta_2 \times similarity_{node}(predicateOf(i), FL) \\ + \theta_3 \times similarity_{node}(objectOf(i), FL) \end{aligned} \tag{7.2}$$

The function $predicateOf(i)$ returns the predicate of the statement i. We compute $similarity_{node}(j, FL)$ where $j \in R \cup SC$ as following:

$$similarity_{node}(j, FL) = \begin{cases} similarity_{type}(\{j\}, FL) & \text{if } j \in SC \\ similarity_{type}(typesOf(j), FL) & \text{if } j \notin SC \end{cases} \tag{7.3}$$

In (7.3), for the case $j \in SC$, we compute the similarity between the class j and the set of classes in FL. For the case $j \notin SC$, we compute the similarity between the set of classes of which j is an instance and the set of classes in FL. The $similarity_{type}$ function takes as arguments a set of classes $TP \subseteq SC$ and the set of filtering criteria FL, and returns the similarity value between them. The $typesOf(j)$ function for a resource $j \in R$ returns the set of classes of which j is an instance. The $similarity_{type}$ function in (7.4a) computes its value by taking the average of all the values of $maxSimilarity_{type}(m, TP)$ where $m \in FL$ and $TP \subseteq SC$. The

[1] http://wimmics.inria.fr/corese

$maxSimilarity_{type}$ function in (7.4b) returns the maximum similarity value between a class m and all the classes in TP. This is to ensure that when a resource is an instance of multiple classes, we filter it by the class which is more similar to the filtering criteria. The $similarity_{type}$ function calculates a combined similarity score of TP with respect to all the classes in FL. Again, we consider the weighted average, and therefore $\sum_i \theta_i = 1$ and $\forall_i : \theta_i \geq 0$ in (7.2).

$$similarity_{type}(TP, FL) = \frac{\sum_{m \in FL} maxSimilarity_{type}(m, TP)}{\mid FL \mid} \tag{7.4a}$$

$$maxSimilarity_{type}(m, TP) = \max_{n \in TP} (similarity_{corese}(m, n)) : \tag{7.4b}$$

For a class $m \in FL$ and a class $n \in TP$, $similarity_{corese}(m, n)$ computes the similarity score between class m and n ranging from 0.0 to 1.0 using SPARQL similarity extension of Corese. A value of 1.0 represent exact match and a value of 0.0 represents completely not similar. The S_{SM} score for a statement indicates the similarity of the information in the statement to the information specified in FL.

7.3.1.3 Abstract Statements

We consider a statement that is close to the root, rs, in corresponding proof tree is more abstract than a statement that is far from the root rs. We define the distance of a node in the proof tree from the root node as the level of the tree to which the node belongs. The root node belongs to level one in the proof tree. The root node is derived from the nodes in level two. A node in level two is derived from the nodes in level three, and so on. For a statement $i \in KST$, we compute the abstraction score $S_{AB}(i)$ using (7.5).

$$S_{AB}(i) = \frac{1}{level(i)} \tag{7.5}$$

The function $level(i)$ returns the proof tree level to which the statement i belongs. We recursively define the function level as follows:

- $level(rs) = 1$
- for every other node i in the proof tree, $level(i) = level(parent(i)) + 1$ where the function $parent(i)$ returns the parent node of i

The $S_{AB}(i)$ measure gives a value greater than 0.0 and less than or equal to 1.0, where a smaller value means less abstract and a larger value means more abstract.

7.3.2 Measures for Re-Ranking

At this point, we can rank the statements of an explanation by combinations of the measures (7.1), (7.2), and (7.5). These measures however do not consider coherence of the information we include in the summaries. Furthermore, they do not consider the importance of the information with respect to their positions in the proof tree. We use two more measures to improve the rankings produced by the combinations of three measures we presented so far. First, we consider the importance of the RDF statements in KST with respect to their proof tree positions. We compute the subtree weight score for a statement i by combining the already computed scores (using combinations of (7.1), (7.2), and (7.5)) of all the statements of the subtree where the statement i is the root. Second, we re-rank already ranked statements by coherence. "Coherence" here means that the ranking of the RDF statements in a summarized explanation should be consistent with respect to their derivations. Our approach is similar to the approach of Zhang *et al.* [Zhang 2007] where they re-rank the RDF statements in an ontology summary after the initial extraction process to satisfy their coherence requirement. Below we describe how we compute the two measures for re-ranking.

7.3.2.1 Subtree Weight in Proof Tree

The salience measure (7.1) indicates the importance of the RDF statements in KST with respect to the oriented graph. But it does not consider the importance of the RDF statements in KST with respect to the proof tree. The idea is to consider a

statement in the proof tree as important if the statements in its subtree are also important. For a subtree of the proof tree with root i, we compute the subtree weight of the statement i by taking the average score of all the statements in that subtree.

$$score_{ST}(i) = \frac{\sum_{j \in subtree(i)} score(j)}{\mid subtree(i) \mid} \tag{7.6}$$

The $subtree(i)$ function returns the RDF statements from the subtree of proof tree with root i. The $score(j)$ for a statement j here can be computed by combinations of the measures we present in section 7.3.1. We discuss more about how to combine the different measures in section 7.4.

7.3.2.2 Coherence

Previous works in text summarization [Eduard 2005] and ontology summarization [Zhang 2007] have shown that coherent information are desirable in summaries. We consider an RDF statement x to be coherent to an RDF statement y if x is directly derived from y. Let RL be a ranked list of RDF statements; S be a list of already selected RDF statements in the summary; i be the next RDF statement to be selected in S. We re-rank RL by repeatedly selecting next i with $\mid RL \mid$ repetitions using (7.7).

$$i = \arg\max_{j \in RL \setminus S} (\lambda_1 \times score(j) + \lambda_2 \times reward(j, S)) \tag{7.7}$$

Again, the $score(j)$ for a statement j here can be computed by combinations of the measures we presented before. We take the weighted average of $score(j)$ and $reward(j, S)$ in (7.7), therefore $\sum_i \lambda_i = 1$ and $\forall_i : \lambda_i \geq 0$.

$$reward(j, S) = 1 - \frac{coherent(S)}{coherent(S \cup j)} \tag{7.8}$$

As (7.8) shows, the *reward* score of a statement j is the amount of potential contribution value – ranging from 0.0 to 1.0 – to the total coherence of the summary if j is added to S. The function $coherent(S)$ in (7.8) returns the number of coherent statements in the summary S. We determine coherence as follows:

- The RDF statement x is coherent to y
  - if $parent(y) = x$

The function $parent(i)$ returns the parent node of i in the proof tree - a node in the prof tree represents an RDF statement.

Note that the starting RDF statement for (7.7) is always the first statement in the ranked list RL in our approach. However, if a different starting RDF statement is selected, then the result of re-ranking by coherence will be different. An interesting idea to explore in future would be to compute different re-ranked list by selecting different starting RDF statements, then from those different re-ranked lists, selecting the best re-ranked list. For this, a cost function to compute the cumulative value of a ranked list would be required.

7.4 Evaluation

Ontology summarization [Li 2010] and text summarization [Eduard 2005, Steinberger 2009] technologies are evaluated by measuring agreements between human-generated summaries – known as "ground truths" – and automatically generated summaries. We obtained our ground truths by surveying 24 people: 17 computer scientists, 1 chemist, 1 social scientist, 1 mathematician, 1 journalist, 1 psychologist, 1 biologist, and 1 business administrator. 18 participants in our survey had knowledge of RDF and 6 participants did not have any knowledge of RDF. The ages of the participants range from 22 to 59. 20 participants were male and 4 were female. The explanations, the questionnaires, the responses, and the results of the

evaluation are publicly available online[2]. We selected a subset of geographical locations from GeoNames[3] and a subset of artists, events, and places from DBPedia[4], then derived new information from these selected subsets. Our ad-hoc reasoner infers new RDF statements with respect to RDFS type propagation; and owl:sameAs and transitivity of the parentFeature property of GeoNames schema. In addition, the reasoner generates explanations for each derivation it performs. We used three test cases – three queries with their results along with the explanations for the results. Each query result is an inferred statement by our reasoner. Each test case has two scenarios: without filtering criteria FL, and with filtering criteria FL. Each participant answered questions for one test case. We randomly assigned a test case to a participant. We asked the participants to rate, from a scale of 1 to 5, the need for each of the statements in the explanation. For, the scenario with filtering criteria FL, we gave the query, the answer, and the explanation but with a user's filtering criteria class taken from the schemata used in the reasoning process. The ratings of the explanation statements are our ground truths. We compute the ground truth rankings of explanation statements by ordering them by their rating values.

7.4.1 Comparing Summarization Measures

We evaluate different combinations of the summarization measures we define. In equation (7.9), we compute $score_{S_{SL}}(i)$ for a statement i considering salience of the statement. We always include S_{SL} in our measure combinations. The motivation is to first include the salient statements in a summary and then find the statements with other measure combination scores (*e.g.* S_{AB} or S_{SM} or $S_{AB} + S_{SM}$) in those salient statements. Equations (7.10), (7.11), and (7.12) show three more measures combinations that we consider for our evaluation. In (7.10), we compute $score_{SL+AB}(i)$ for a statement i considering salience and abstractness of the statement. In (7.11), we compute the $score_{SL+SM}(i)$ for a statement i considering

[2]http://ns.inria.fr/ratio4ta/sm/
[3]http://www.geonames.org/
[4]http://dbpedia.org/

the salience (S_{SL}), and the similarity (S_{SM}) with respect to user's filtering criteria FL. In (7.12), we compute $score_{SL+AB+SM}(i)$ for a statement i considering the salience (S_{SL}), the abstractness (S_{AB}), and the similarity (S_{SM}) with respect to user's filtering criteria FL.

$$score_{SL}(i) = S_{SL}(i) \tag{7.9}$$

$$score_{SL+AB}(i) = \lambda_1 \times S_{SL}(i) + \lambda_2 \times S_{AB}(i) \tag{7.10}$$

$$score_{SL+SM}(i) = \lambda_1 \times S_{SL}(i) + \lambda_2 \times S_{SM}(i, FL) \tag{7.11}$$

$$\begin{aligned} score_{SL+AB+SM}(i) = \lambda_1 \times S_{SL}(i) + \lambda_2 \times S_{AB}(i) \\ + \lambda_3 \times S_{SM}(i, FL) \end{aligned} \tag{7.12}$$

These combinations are combinations of ranking measures we present in section 7.3.1. For re-ranking, we first compute the score using any of (7.9), (7.10), (7.11), and (7.12), then we re-rank using (7.6), or (7.7). In remaining of this chapter, we denote subtree weight measure as S_{ST}, and coherence measure as S_{CO}. For the scenario without FL, we compare our summaries to sentence graph summarization [Zhang 2007] – denoted as S_{SG}. As the authors of sentence graph summarization approach suggest, we use 0.8 as the navigational preference p parameter value. Zhang *et al.* use navigational preference to determine the weight of links between RDF sentences during the summarization process. We implemented sentence graph summarization using degree centrality as the authors found degree centrality performs better than other centrality measures in general, and for its simplicity. We do not consider sentence graph summarization for scenarios with FL because sentence graph summarization does not have a feature for filtering information using ontology concepts as filtering criteria.

In (7.10), (7.11), and (7.12), $\sum_i \lambda_i = 1$ and $\forall_i : \lambda_i \geq 0$. Thus we take the weighted averages of the measure combinations. For this evaluation, we use equal weights in (7.10), (7.11), (7.12), (7.1), (7.2), and (7.7). Therefore, we set $\forall_i : \lambda_i = \frac{1}{N_\lambda}$ in (7.10), (7.11), (7.12), and (7.7) where N_λ = number of λ parameters in the

	avg.	std. dev.
Without FL	0.836	0.048
With FL	0.835	0.065

Table 7.1: Average agreements between ratings measured by cosine similarity.

corresponding equations; and $\forall_i : \theta_i = \frac{1}{N_\theta}$ in (7.1), and (7.2) where N_θ = number of θ parameters in the corresponding equations. However, one can use parameter estimation techniques for finding the optimal parameter values.

7.4.2 Analysis of Ground Truths

We use cosine similarity to measure the agreements between rating vectors. Cosine similarity values in positive space are in the interval 0 to 1. Table 7.1 shows the total average agreement measured by cosine similarity and standard deviations for two scenarios – without filtering criteria FL and with filtering criteria FL. The average agreements for both the scenarios are more than 0.8 which is considerably high. However, the standard deviation is higher for the scenario with FL. The reason for this higher standard deviation is that the participants had to consider the highly subjective [Araújo 2007] factor of similarity and therefore their ratings had more variance for the scenario with FL.

7.4.3 Evaluating the Rankings

We use normalized discounted cumulative gain to evaluate ranking quality. Discounted cumulative gain (DCG) [Järvelin 2002, McSherry 2008] measures the quality of results of an information retrieval system in a ranked list. DCG assumes that judges have graded each item in a list of results. Using these grades, DCG measures the usefulness, or gain, of a ranked list of results. DCG penalizes high quality results appearing lower in a ranked list of results. Normalized discounted cumulative Gain ($nDCG$) allows to calculate and compare this measure across multiple lists of results where each of the lists might have different length. $nDCG$ values are in the interval

0.0 to 1.0. An $nDCG_p$ value of 1.0 means that the ranking is perfect at position p with respect to the ideal ranking – ranking based on grades. The $nDCG_p$ value 0.0 means that the ranking is completely imperfect at position p with respect to the ideal ranking. In our study, the average of ratings by all the survey participants for a statement s is the grade for the statement s. Figure 7.2 shows the average $nDCG$ values of the three test cases for different rankings by different measure combinations. The *x-axis* represents ranks and the *y-axis* represents $nDCG$. We plot 21 ranks in the *x-axis* because the shortest explanation among the three test cases had 21 statements. For the scenario without FL (the figure on the left), the measure combinations $S_{SL} + S_{AB} + S_{CO}$, $S_{SL} + S_{AB} + S_{ST}$, and $S_{SL} + S_{AB} + S_{ST} + S_{CO}$ produce closer rankings to the ground truth rankings. For the scenario with FL (the figure on the right), the same three measure combinations with added S_{SM} measure have the best $nDCG$ values. This means that the participants consider central (with respect to the oriented graph and the proof tree), abstract, and coherent information as necessary information in explanation summaries for the scenario without FL. This also holds for the scenario with FL with the added observation that the participants also consider similar information as necessary information. The $nDCG$ values for these measure combinations are higher than 0.9 for all ranks. This means that the rankings by these measure combinations are highly similar to the ground truth rankings. In contrast, the sentence graph summarization ranking has low $nDCG$ values compared to all the other rankings for the scenario without FL. This shows that our explanation summarization algorithms produce much higher quality rankings than sentence graph summarization algorithm.

7.4.4 Evaluating the Summaries

We evaluate the summaries using *Recall* and *Precision* composite scores as in text summarization [Eduard 2005]. *Recall* and *Precision* quantify how closely the algorithm generated summaries correspond to the human produced summaries. *Recall*

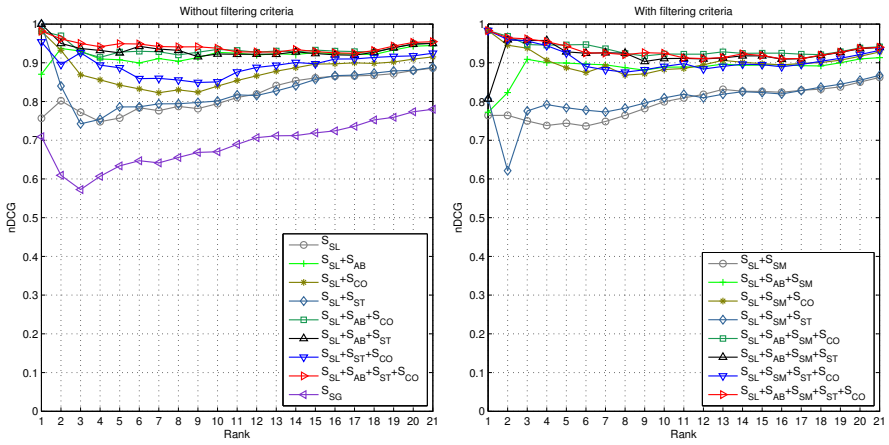


Figure 7.2: Comparison of rankings.

reflects how many good statements the algorithm missed, and *Precision* reflects how many of the algorithm's selected statements are good. *F-score* is the composite measure of *Recall* and *Precision*. We use the basic *F-score* as in [Steinberger 2009]: $F\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall}$. We measure *F-score* for summarized explanations with different compression ratios, CR, to evaluate summaries of different sizes. Compression ratio CR is the ratio of the size of the summarized explanation to the size of original explanation. We evaluate the summarized explanations produced by different measure combinations by comparing them to human generated summarized explanations (*i.e.* ground truth summarized explanations) using *F-score*. To generate the ground truth summarized explanation for an explanation, we include a statement in the ground truth summarized explanation if its rating is greater than or equal to the average rating of all the statements in the original explanation. *F-scores* reflects the accuracy of automatically generated summaries with respect to the ground truth summary. A desirable situation would be a summarized explanation with high *F-score* and low CR. Figure 7.3 shows the average *F-scores* for different measure combinations for summaries with different sizes for the three test cases. The *x-axis* represents compression ratio CR. The *y-axis* represents *F-scores*. For the scenario without FL (the figure on the left), the best *F-score* is 0.72 when CR value is 0.33 by the measure combinations $S_{SL} + S_{AB} + S_{ST}$ and $S_{SL} + S_{AB} + S_{ST} + S_{CO}$. This is a desirable situation with a high *F-score* and low CR. The sentence graph summa-

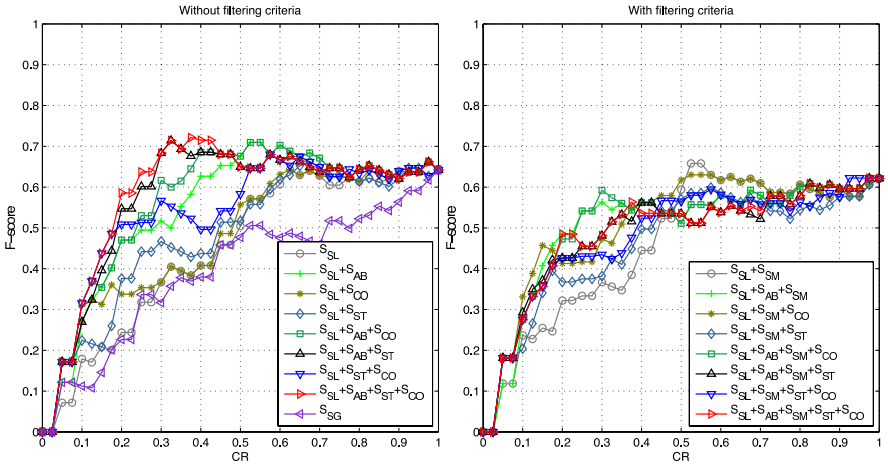


Figure 7.3: Compression ratio (CR) vs *F-score*.

rization performs poorly with a best *F-score* value of 0.34 in the CR interval 0.05 to 0.3. This shows that our summarized explanations are more accurate than the summarized explanations generated by sentence graph summarization algorithm. For the scenario with FL (the figure on the right), the best *F-score* is 0.66 at CR values 0.53 and 0.55 by the measure combination$S_{SL} + S_{SM}$. However, the *F-score* 0.6 at CR value 0.3 by the measure combination $S_{SL} + S_{AB} + S_{SM} + S_{CO}$ is more desirable because the size of the summary is smaller. As expected, our summarization approach perform worse in the scenario with FL where we use S_{SM}. This is due to the fact that the survey participants had to consider the highly subjective factor of similarity.

7.5 Summary

In this chapter, we presented five measures to summarize Linked Explanations. We evaluate different combinations of these measures. The evaluation shows that our approach produces high quality rankings for summarizing explanation statements. Our summarized explanations are also highly accurate with *F-score* values ranging from 0.6 to 0.72 for small summaries. Our approach outperforms the sentence graph based ontology summarization approach.

Chapter 8

Conclusion and Perspectives

Contents

8.1 Summary of Contributions

In this thesis, we aim at assisting users in understanding query behavior and results in the context of consuming Linked Data. We have contributions in five areas: query performance prediction, query result provenance, evaluating explanations, explanation for Linked Data, and summarizing explanations for Linked Data.

Query Performance Prediction. Existing approaches for SPARQL query cost estimation are based on statistics about the underlying data. However, statistics about the underlying data are often missing in Linked Data. We present a machine learning approach to predict query performance metrics. We learn query execution times from already executed queries – without using statistics about the underlying RDF data. We discuss how to model SPARQL queries as feature vectors, and show highly accurate predictions. Predicted query performance metrics using our approach can be used to assist users to understand

query performance for workload management related tasks to meet specific QoS targets in the context of querying Linked Data.

Query Result Provence. Previous works on generating *why-provenance* for SPARQL query results are based on what is know as the annotation approach (the eager approach) where the underlying data model, the query language, and the query processing engine are re-engineered to compute provenance during the query processing. However, re-engineering the underlying data model, the query language, or the query processor is often not possible in the Linked Data scenario. We present a non-annotation approach to generate *why-provenance* for SPARQL query results and show its feasibility for common Linked Data queries. We generate the explanation for a SPARQL query result tuple from its *why-provenance*. We present an explanation-aware federated query processor prototype and show the presentations of our explanations.

Evaluating Explanations. Previous works on explanations in the Semantic Web literature work on the assumptions that explanations would improve users' understanding and trust. However, previous works do not evaluate such assumptions. We present a user study to evaluate the impact of query result explanations in a federated query processing scenario for Linked Data. Our user study shows that our query result explanations are helpful for end users to understand the result derivations and make trust judgments on the results.

Explanations for Linked Data. Much of the previous work on explanations for the Semantic Web do not address explanation in a distributed environment. The Inference Web [McGuinness 2003] approach proposes a centralized registry based solution for publishing explanation metadata from distributed reasoners. In contrast, we propose a decentralized solution to this problem. We discuss how to represent and generate explanations for Linked Data. We present the *Ratio4TA* vocabulary to describe explanation metadata and in-

troduce the notion of Linked Explanations – publishing explanation metadata as Linked Data. This enables explaining distributed data in a decentralized fashion. *Ratio4TA* extends the W3C PROV Ontology to enable data consumers to process explanation metadata according to W3C PROV standards. We also show how to generate natural language based explanations from these explanation metadata.

Summarizing Explanations for Linked Data. Although explanations with the details of all the derivation steps may be useful for expert users, they may overwhelm non-expert users with too much information. In addition, an expert user such as a knowledge engineer may want to focus on a specific part of a detailed explanation. A knowledge engineer may also want a short explanation to have an overview of the reasoning. We presented five measures to summarize explanations. We evaluate different combinations of these measures. The evaluation shows that our approach produces high quality rankings for summarizing explanation statements. Our summarized explanations are highly accurate with *F-score* values ranging from 0.6 to 0.72 for small summaries. Our approach outperforms the sentence graph based ontology summarization approach.

8.2 Perspectives

We have several perspectives for our query performance prediction, query result explanation, and Linked Explanations approaches.

8.2.1 Query Performance Prediction

In future, firstly we would like to use our approach in query optimization and compare it to traditional query cost estimation techniques in the Linked Data scenario – e.g. join order optimization in federated query processing. State of the art Linked Data query processing approach FedX [Schwarte 2011] uses variable count selectivity

estimation [Stocker 2008] optimization for efficient join ordering of grouped triple pattern execution. We would like to compare our approach to such approaches. Second, we plan to systematically generate training queries for two scenarios: (a) given query logs of real queries (b) given a small set of sample queries. We plan to apply query log mining techniques to systematically generate training queries. Recent work [Arias 2011] on query log mining shows that the majority of SPARQL queries share some common characteristics. We plan to consider those statistically significant common characteristics in refining training queries from massive query logs and generating training queries from a small set of sample queries. We would also explore how these common characteristics can be used as query features. Third, we would like to investigate online machine learning techniques for our models. Our goal would be to refine our prediction models based on the new predictions and their actual values. Finally, we would like to include load and availability related features. In this direction, we plan to execute the training queries every hour and include features such as time, day, and month. This would help us to model workload patterns for public SPARQL endpoints.

8.2.2 Query Result Explanation

In the future work, we would like to extend our algorithm to generate *how-provenance*, which explain how a result tuple was derived with the details of the operations performed in the derivation. The performed SPARQL operations can be extracted from the query patterns of SPARQL queries the same way we extract the *why-provenance* triples. In fact, the algebraic expression tree we generate during the *why-provenance* extraction process already contains these operations. For *how-provenance*, we would have to associate these operators to the extracted *why-provenance* triples. Furthermore, currently we present the first derivation in a *why-provenance* as explanation in our explanation user interface. It would be interesting to explore how we can effectively present information from *why-provenance* as ex-

planations to users. In this direction, one approach could be to rank the derivations of *why-provenance*, which would require us to define ranking criteria for derivations of *why-provenance*. Finally, our user study to evaluate the impact of query result explanations had only 11 participants. The participants needed to have some notions of RDF and SPARQL, and be motivated to simulate a simple federated query solution process. Although we went through the prominent communication channels (mailing lists and twitter hastags), it was difficult to find a large number of participants. In addition, as the participants were anonymized, we could not go back to the participants and ask why a given participant has provided a given answer to re-evaluate their choices. A controlled user study with a large number of participants would provide us more conclusive results and re-evaluate the choices of the participants. One approach to conduct such a controlled user study would to use a crowdsourcing infrastructure such as Amazon's Mechanical Turk[1] where participants would be provided financial incentives for their participations.

8.2.3 Linked Explanations and Summarization

As we discuss in Chapter 6, our Linked Explanations approach requires the data triples to be reifiable. We use named graphs for reifying data triples and group together explanation metadata triples. Currently the best practices for publishing named data as Linked Data has not been universally agreed upon by the Semantic Web community [Shinavier 2010]. However, following the adoption of named graphs in RDF 1.1, it is expected that there would be a community consensus on best practices for publishing named graphs as Linked Data. Furthermore, the amount of explanation related metadata in our approach can become very large. Therefore, efficient and scalable storage and querying techniques would be required to use our approach in practice. In this direction, there is a large literature on scalable storage, indexing, and querying for RDF [Hose 2011]. These existing approaches can be used to store and serve the large amount of explanation related metadata. Finally, we

[1] https://www.mturk.com/

would like to explore how we can effectively present explanations and summarized explanations using different kinds of user interfaces and user interactions. We would like to explore how we can effectively use the summarization rankings while presenting information – e.g not expanding a proof tree branch which contains statements with low ranking scores.

Appendix A

Ratio4TA Vocabulary

We present the source code of *Ratio4TA* vocabulary below using Turtle notation.

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix rdfg: <http://www.w3.org/2004/03/trix/rdfg-1/> .
@prefix ns: <http://www.w3.org/ns/> .
@prefix : <http://ns.inria.fr/ratio4ta/v3#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://ns.inria.fr/ratio4ta/v3> .

<http://ns.inria.fr/ratio4ta/v3> rdf:type owl:Ontology ;
                                 rdfs:comment "Ratio4TA (interlinked justifications for triple assertions) is a lightweight
                                       vocabulary for encoding justifications using named graphs."@en .

#################################
# Annotation properties
#################################
prov:unqualifiedForm rdf:type owl:AnnotationProperty .
prov:aq rdf:type owl:AnnotationProperty .
prov:prov-n rdf:type owl:AnnotationProperty .
prov:sharesDefinitionWith rdf:type owl:AnnotationProperty .
prov:prov-dm rdf:type owl:AnnotationProperty .
prov:definition rdf:type owl:AnnotationProperty .
prov:editorialNote rdf:type owl:AnnotationProperty .
prov:inverse rdf:type owl:AnnotationProperty .
prov:constraints rdf:type owl:AnnotationProperty .
prov:dm rdf:type owl:AnnotationProperty .
prov:category rdf:type owl:AnnotationProperty .
prov:prov-dm-constraints rdf:type owl:AnnotationProperty .
prov:editorsDefinition rdf:type owl:AnnotationProperty .
prov:component rdf:type owl:AnnotationProperty .
prov:agent rdf:type owl:AnnotationProperty ;
           rdfs:label "agent" ;
           prov:component "alternate" ;
           prov:inverse "agentOfInfluence" ;
           prov:editorialNote "This property behaves in spirit like rdf:object; it references the object of a prov:involved
                 triple." ;
           rdfs:comment "The property used by a prov:AgentInvolvement to cite the Agent that was prov:involved with either an
                  Activity or Entity. It can be used to express the agent involved in being responsible for an activity,
```

```
                being attributed to an entity, starting or ending an activity, or being responsible for another subordinate
                 agent in an activity."@en ;
38          prov:editorsDefinition "The prov:agent property references an prov:Agent which influenced a resource. This
                property applies to an prov:AgentInfluence, which is given by a subproperty of prov:qualifiedInfluence from
                 the influenced prov:Entity, prov:Activity or prov:Agent."@en ;
39          prov:inverse "agentInvolvement" ;
40          prov:editorialNote "This property behaves in spirit like rdf:object; it references the object of a prov:
                wasInfluencedBy triple."@en ;
41          prov:category "qualified" ;
42          rdfs:isDefinedBy ns:prov-o# .
43  prov:qualifiedForm rdf:type owl:AnnotationProperty .
44  prov:n rdf:type owl:AnnotationProperty .
45
46  ###################################
47  # Object Properties
48  ###################################
49  ### http://ns.inria.fr/ratio4ta/v3#belongsTo
50  :belongsTo rdf:type owl:ObjectProperty ;
51          rdfs:label "belongs to"@en ;
52          prov:definition "A reasoning process uses input data and computes results. Each of these computed results includes
                output data. The belongsTo property specifies an output data belongs to a result."@en ;
53          rdfs:domain :OutputData ;
54          rdfs:range :Result ;
55          owl:inverseOf :contains ;
56          rdfs:subPropertyOf prov:qualifiedGeneration .
57
58  ### http://ns.inria.fr/ratio4ta/v3#computed
59  :computed rdf:type owl:ObjectProperty ;
60          rdfs:label "computed"@en ;
61          prov:definition "A reasoning process uses input data and computes results. Each of these computed results includes
                output data. The contains property specifies a result contains an output data."@en ;
62          rdfs:domain :ReasoningProcess ;
63          rdfs:range :Result ;
64          rdfs:subPropertyOf prov:qualifiedAssociation .
65
66  ### http://ns.inria.fr/ratio4ta/v3#contains
67  :contains rdf:type owl:ObjectProperty ;
68          rdfs:label "contains"@en ;
69          prov:definition "Specifies the output data contained in a result."@en .
70
71  ### http://ns.inria.fr/ratio4ta/v3#derivationReasoner
72  :derivationReasoner rdf:type owl:ObjectProperty ;
73                  rdfs:label "has derivation reasoner"@en ;
74                  prov:definition "A software application performs data derivations. The derivationReasoner property
                        specifies a data derivation performed by a software application."@en ;
75                  rdfs:domain :DataDerivation ;
76                  rdfs:range :SoftwareApplication ;
77                  rdfs:subPropertyOf prov:agent .
78
79  ### http://ns.inria.fr/ratio4ta/v3#derivedBy
80  :derivedBy rdf:type owl:ObjectProperty ;
81          rdfs:label "derived by"@en ;
82          prov:definition "A derivation uses rules and derives output data. The derivedBy property specifies an output data
                derived by a derivation."@en ;
83          rdfs:range :DataDerivation ;
```

```
84            rdfs:domain :OutputData ;
85            rdfs:subPropertyOf prov:qualifiedDerivation .
86
87    ### http://ns.inria.fr/ratio4ta/v3#derivedFrom
88    :derivedFrom rdf:type owl:ObjectProperty ;
89              rdfs:label "derived from"@en ;
90              prov:definition "A derivation transforms a data into another, constructs a data into another, or updates a data,
                      resulting in a new one. Note that by data we mean an instance of the Data class. The derivedFrom property
                      specifies a data derived from a data."@en ;
91              rdfs:domain :Data ;
92              rdfs:range :Data ;
93              rdfs:subPropertyOf prov:wasDerivedFrom .
94
95    ### http://ns.inria.fr/ratio4ta/v3#explains
96    :explains rdf:type owl:ObjectProperty ;
97            rdfs:label "explains"@en ;
98            prov:definition "An ExplanationBundle contains explanation statements for a Data. The explains property specifies
                  an ExplanationBundle that explains a data."@en ;
99            rdfs:range :Data ;
100           rdfs:domain :ExplanationBundle .
101
102   ### http://ns.inria.fr/ratio4ta/v3#hasExplanation
103   :hasExplanation rdf:type owl:ObjectProperty ;
104               rdfs:label "has explanation"@en ;
105               prov:definition "An ExplanationBundle contains explanation statements for a Data. The hasExplanation property
                      specifies a data explained by an ExplanationBundle."@en ;
106               rdfs:domain :Data ;
107               rdfs:range :ExplanationBundle ;
108               owl:inverseOf :explains ;
109               rdfs:subPropertyOf prov:has_provenance .
110
111   ### http://ns.inria.fr/ratio4ta/v3#performedAsPartOf
112   :performedAsPartOf rdf:type owl:ObjectProperty ;
113                 rdfs:label "performed as part of"@en ;
114                 prov:definition "A reasoning process performs data derivations. The performedAsPartOf specifies a data
                        derivation is performed as part of a reasoning process."@en ;
115                 rdfs:domain :DataDerivation ;
116                 rdfs:range :ReasoningProcess ;
117                 rdfs:subPropertyOf prov:hadActivity .
118
119   ### http://ns.inria.fr/ratio4ta/v3#performedBy
120   :performedBy rdf:type owl:ObjectProperty ;
121             rdfs:label "performed by"@en ;
122             prov:definition "A software application performs a reasoning process. The performedBy property specifies a
                    reasoning process is performed by a software application."@en ;
123             rdfs:domain :ReasoningProcess ;
124             rdfs:range :SoftwareApplication ;
125             rdfs:subPropertyOf prov:wasAssociatedWith .
126
127   ### http://ns.inria.fr/ratio4ta/v3#produced
128   :produced rdf:type owl:ObjectProperty ;
129           rdfs:label "produced"@en ;
130           prov:definition "A reasoning process computes results and a computed result contains output data. The produced
                property specifies a reasoning process produced an instance of data."@en ;
131           rdfs:range :OutputData ;
```

```
132          rdfs:domain :ReasoningProcess ;
133          rdfs:subPropertyOf prov:generated .
134
135  ### http://ns.inria.fr/ratio4ta/v3#resultReasoner
136  :resultReasoner rdf:type owl:ObjectProperty ;
137                rdfs:label "has result reasoner"@en ;
138                prov:definition "A software application performs reasoning to computes results. The resultReasoner property
                         specifies result has an associated software application."@en ;
139                rdfs:domain :Result ;
140                rdfs:range :SoftwareApplication ;
141                rdfs:subPropertyOf prov:agent .
142
143  ### http://ns.inria.fr/ratio4ta/v3#summarizationOf
144  :summarizationOf rdf:type owl:ObjectProperty ;
145                 rdfs:label "summarization of"@en ;
146                 prov:definition "A summarized justification can contain the most important information from several
                        justifications. The summarizationOf property specifies a justification account is a summarization of a
                         justification account. Since a summarized justification account is a summary of multiple
                        justification accounts, there will be multiple statements describing the links between a summarized
                        justification account and its original justification accounts using this property."@en ;
147                 rdfs:range :ExplanationBundle ;
148                 rdfs:domain :ExplanationBundle ;
149                 rdfs:subPropertyOf prov:generalizationOf .
150
151  ### http://ns.inria.fr/ratio4ta/v3#usedData
152  :usedData rdf:type owl:ObjectProperty ;
153          rdfs:label "used data"@en ;
154          prov:definition "A reasoning process uses input data to compute its results. The usedData property specifies a
                 reasoning process used an instance of input data."@en ;
155          rdfs:range :InputData ;
156          rdfs:domain :ReasoningProcess ;
157          rdfs:subPropertyOf prov:used .
158
159  ### http://ns.inria.fr/ratio4ta/v3#usedRule
160  :usedRule rdf:type owl:ObjectProperty ;
161          rdfs:label "used rule"@en ;
162          prov:definition "Data derivations use rules to perform derivations. The usedRule property specifies a data
                 derivation used a rule."@en ;
163          rdfs:domain :DataDerivation ;
164          rdfs:range :Rule ;
165          rdfs:subPropertyOf prov:hadPlan .
166
167  ### http://ns.inria.fr/ratio4ta/v3#wasInvolvedComputing
168  :wasInvolvedComputing rdf:type owl:ObjectProperty ;
169                      rdfs:label "was involved in computing"@en ;
170                      prov:definition "A reasoning process performs data derivations to compute results. The
                             wasInvolvedComputing property specifies a data derivation was involved in computing a result."@en
                              ;
171                      rdfs:domain :DataDerivation ;
172                      rdfs:range :Result ;
173                      rdfs:subPropertyOf prov:hadGeneration .
174
175  ### http://www.w3.org/ns/prov#activity
176  prov:activity rdf:type owl:ObjectProperty ;
177               rdfs:label "activity" ;
```

```
178             prov:editorsDefinition "The prov:activity property references an prov:Activity which influenced a resource.
                      This property applies to an prov:ActivityInfluence, which is given by a subproperty of prov:
                      qualifiedInfluence from the influenced prov:Entity, prov:Activity or prov:Agent." ;
179             prov:inverse "activityOfInfluence" ;
180             prov:editorialNote "This property behaves in spirit like rdf:object; it references the object of a prov:
                      wasInfluencedBy triple."@en ;
181             prov:category "qualified" ;
182             rdfs:range prov:Activity ;
183             rdfs:domain prov:ActivityInfluence ;
184             rdfs:subPropertyOf prov:influencer ;
185             rdfs:isDefinedBy ns:prov-o# .
186
187 ### http://www.w3.org/ns/prov#agent
188 prov:agent rdf:type owl:ObjectProperty ;
189           rdfs:label "agent" ;
190           prov:editorialNote "This property behaves in spirit like rdf:object; it references the object of a prov:involved
                triple." ;
191           rdfs:comment "The property used by a prov:AgentInvolvement to cite the Agent that was prov:involved with either an
                 Activity or Entity. It can be used to express the agent involved in being responsible for an activity,
                being attributed to an entity, starting or ending an activity, or being responsible for another subordinate
                 agent in an activity."@en ;
192           prov:category "qualified" ;
193           prov:component "alternate" ;
194           prov:editorsDefinition "The prov:agent property references an prov:Agent which influenced a resource. This
                property applies to an prov:AgentInfluence, which is given by a subproperty of prov:qualifiedInfluence from
                 the influenced prov:Entity, prov:Activity or prov:Agent."@en ;
195           prov:inverse "agentOfInfluence" ,
196                        "agentInvolvement" ;
197           prov:editorialNote "This property behaves in spirit like rdf:object; it references the object of a prov:
                wasInfluencedBy triple."@en ;
198           rdfs:range prov:Agent ;
199           rdfs:domain prov:AgentInfluence ;
200           rdfs:subPropertyOf prov:influencer ;
201           rdfs:isDefinedBy ns:prov-o# .
202
203 ### http://www.w3.org/ns/prov#alternateOf
204 prov:alternateOf rdf:type owl:ObjectProperty ;
205                 rdfs:label "alternateOf" ;
206                 prov:constraints "http://www.w3.org/TR/2013/REC-prov-constraints-20130430/#prov-dm-constraints-fig"^^xsd:
                        anyURI ;
207                 prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-alternate"^^xsd:anyURI ;
208                 prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-alternate"^^xsd:anyURI ;
209                 prov:definition "Two alternate entities present aspects of the same thing. These aspects may be the same or
                        different, and the alternate entities may or may not overlap in time."@en ;
210                 prov:category "expanded" ;
211                 prov:component "alternate" ;
212                 prov:inverse "alternateOf" ;
213                 rdfs:isDefinedBy prov: ;
214                 rdfs:domain prov:Entity ;
215                 rdfs:range prov:Entity ;
216                 owl:inverseOf prov:alternateOf ;
217                 rdfs:seeAlso prov:specializationOf ;
218                 rdfs:isDefinedBy ns:prov-o# .
219
220 ### http://www.w3.org/ns/prov#entity
```

```
prov:entity rdf:type owl:ObjectProperty ;
           rdfs:label "entity" ;
           prov:editorsDefinition "The prov:entity property references an prov:Entity which influenced a resource. This
                 property applies to an prov:EntityInfluence, which is given by a subproperty of prov:qualifiedInfluence
                 from the influenced prov:Entity, prov:Activity or prov:Agent." ;
           prov:inverse "entityOfInfluence" ;
           prov:editorialNote "This property behaves in spirit like rdf:object; it references the object of a prov:
                 wasInfluencedBy triple."@en ;
           prov:category "qualified" ;
           rdfs:range prov:Entity ;
           rdfs:domain prov:EntityInfluence ;
           rdfs:subPropertyOf prov:influencer ;
           rdfs:isDefinedBy ns:prov-o# .

### http://www.w3.org/ns/prov#generalizationOf
prov:generalizationOf rdf:type owl:ObjectProperty ;
                    rdfs:label "generalizationOf" ;
                    rdfs:isDefinedBy prov: ;
                    owl:inverseOf prov:specializationOf .

### http://www.w3.org/ns/prov#generated
prov:generated rdf:type owl:ObjectProperty ;
              rdfs:label "generated" ;
              prov:component "entities-activities" ;
              prov:inverse "wasGeneratedBy" ;
              prov:category "expanded" ;
              prov:editorialNote "prov:generated is one of few inverse property defined, to allow Activity-oriented
                    assertions in addition to Entity-oriented assertions."@en ;
              rdfs:isDefinedBy prov: ;
              rdfs:domain prov:Activity ;
              rdfs:range prov:Entity ;
              prov:sharesDefinitionWith prov:Generation ;
              rdfs:subPropertyOf prov:influenced ;
              rdfs:isDefinedBy ns:prov-o# .

### http://www.w3.org/ns/prov#hadActivity
prov:hadActivity rdf:type owl:ObjectProperty ;
                rdfs:label "hadActivity" ;
                rdfs:comment "This property has multiple RDFS domains to suit multiple OWL Profiles. See <a href=\"#owl-
                      profile\">PROV-O OWL Profile</a>." ,
                           "The _optional_ Activity of an Influence, which used, generated, invalidated, or was the
                                 responsibility of some Entity. This property is _not_ used by ActivityInfluence (use prov:
                                 activity instead)."@en ;
                prov:editorialNote "The multiple rdfs:domain assertions are intended. One is simpler and works for OWL-RL,
                      the union is more specific but is not recognized by OWL-RL."@en ;
                prov:component "derivations" ;
                prov:category "qualified" ;
                prov:inverse "wasActivityOfInfluence" ;
                rdfs:range prov:Activity ;
                prov:sharesDefinitionWith prov:Activity ;
                rdfs:domain prov:Influence ;
                rdfs:isDefinedBy ns:prov-o# ;
                rdfs:domain [ rdf:type owl:Class ;
                            owl:unionOf ( prov:Delegation
                                        prov:Derivation
```

```
                                    prov:End
                                    prov:Start
                                  )
                        ] .

### http://www.w3.org/ns/prov#hadGeneration
prov:hadGeneration rdf:type owl:ObjectProperty ;
                rdfs:label "hadGeneration" ;
                prov:inverse "generatedAsDerivation" ;
                prov:category "qualified" ;
                rdfs:comment "The _optional_ Generation involved in an Entity's Derivation."@en ;
                prov:component "derivations" ;
                rdfs:domain prov:Derivation ;
                prov:sharesDefinitionWith prov:Generation ;
                rdfs:range prov:Generation ;
                rdfs:isDefinedBy ns:prov-o# .

### http://www.w3.org/ns/prov#hadPlan
prov:hadPlan rdf:type owl:ObjectProperty ;
            rdfs:label "hadPlan" ;
            prov:category "qualified" ;
            prov:component "agents-responsibility" ;
            prov:inverse "wasPlanOf" ;
            rdfs:comment "The _optional_ Plan adopted by an Agent in Association with some Activity. Plan specifications are
                   out of the scope of this specification."@en ;
            rdfs:domain prov:Association ;
            prov:sharesDefinitionWith prov:Plan ;
            rdfs:range prov:Plan ;
            rdfs:isDefinedBy ns:prov-o# .

### http://www.w3.org/ns/prov#has_provenance
prov:has_provenance rdf:type owl:ObjectProperty ;
                  rdfs:label "has_provenance" ;
                  prov:aq "http://www.w3.org/TR/2013/NOTE-prov-aq-20130430/#resource-represented-as-html"^^xsd:anyURI ;
                  prov:inverse "provenanceOf" ;
                  rdfs:comment "Indicates a provenance-URI for a resource; the resource identified by this property
                        presents a provenance record about its subject or anchor resource."@en ;
                  prov:category "access-and-query" ;
                  rdfs:isDefinedBy prov: .

### http://www.w3.org/ns/prov#influenced
prov:influenced rdf:type owl:ObjectProperty ;
               rdfs:label "influenced" ;
               prov:inverse "wasInfluencedBy" ;
               prov:component "agents-responsibility" ;
               prov:category "expanded" ;
               rdfs:isDefinedBy prov: ;
               prov:sharesDefinitionWith prov:Influence ;
               rdfs:isDefinedBy ns:prov-o# .

### http://www.w3.org/ns/prov#influencer
prov:influencer rdf:type owl:ObjectProperty ;
               rdfs:label "influencer" ;
               prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-influence"^^xsd:anyURI ;
               prov:category "qualified" ;
```

```
321             prov:inverse "hadInfluence" ;
322             rdfs:comment "Subproperties of prov:influencer are used to cite the object of an unqualified PROV-O triple
                    whose predicate is a subproperty of prov:wasInfluencedBy (e.g. prov:used, prov:wasGeneratedBy). prov:
                    influencer is used much like rdf:object is used."@en ;
323             prov:editorialNote "This property and its subproperties are used in the same way as the rdf:object property,
                    i.e. to reference the object of an unqualified prov:wasInfluencedBy or prov:influenced triple."@en ;
324             prov:editorsDefinition "This property is used as part of the qualified influence pattern. Subclasses of prov:
                    Influence use these subproperties to reference the resource (Entity, Agent, or Activity) whose
                    influence is being qualified."@en ;
325             rdfs:range owl:Thing ;
326             rdfs:domain prov:Influence ;
327             rdfs:isDefinedBy ns:prov-o# .
328
329 ### http://www.w3.org/ns/prov#qualifiedAssociation
330 prov:qualifiedAssociation rdf:type owl:ObjectProperty ;
331                       rdfs:label "qualifiedAssociation" ;
332                       prov:inverse "qualifiedAssociationOf" ;
333                       rdfs:comment "If this Activity prov:wasAssociatedWith Agent :ag, then it can qualify the Association
                              using prov:qualifiedAssociation [ a prov:Association; prov:agent :ag; :foo :bar ]."@en ;
334                       prov:component "agents-responsibility" ;
335                       prov:category "qualified" ;
336                       rdfs:domain prov:Activity ;
337                       rdfs:range prov:Association ;
338                       prov:sharesDefinitionWith prov:Association ;
339                       rdfs:subPropertyOf prov:qualifiedInfluence ;
340                       prov:unqualifiedForm prov:wasAssociatedWith ;
341                       rdfs:isDefinedBy ns:prov-o# .
342
343 ### http://www.w3.org/ns/prov#qualifiedDerivation
344 prov:qualifiedDerivation rdf:type owl:ObjectProperty ;
345                      rdfs:label "qualifiedDerivation" ;
346                      prov:component "derivations" ;
347                      prov:category "qualified" ;
348                      rdfs:comment "If this Entity prov:wasDerivedFrom Entity :e, then it can qualify how it was derived
                             using prov:qualifiedDerivation [ a prov:Derivation; prov:entity :e; :foo :bar ]."@en ;
349                      prov:inverse "qualifiedDerivationOf" ;
350                      prov:sharesDefinitionWith prov:Derivation ;
351                      rdfs:range prov:Derivation ;
352                      rdfs:domain prov:Entity ;
353                      rdfs:subPropertyOf prov:qualifiedInfluence ;
354                      prov:unqualifiedForm prov:wasDerivedFrom ;
355                      rdfs:isDefinedBy ns:prov-o# .
356
357 ### http://www.w3.org/ns/prov#qualifiedGeneration
358 prov:qualifiedGeneration rdf:type owl:ObjectProperty ;
359                      rdfs:label "qualifiedGeneration" ;
360                      prov:inverse "qualifiedGenerationOf" ;
361                      prov:component "entities-activities" ;
362                      prov:category "qualified" ;
363                      rdfs:comment "If this Activity prov:generated Entity :e, then it can qualify how it performed the
                             Generation using prov:qualifiedGeneration [ a prov:Generation; prov:entity :e; :foo :bar ]."@en
                              ;
364                      rdfs:domain prov:Entity ;
365                      rdfs:range prov:Generation ;
366                      prov:sharesDefinitionWith prov:Generation ;
```

```
367                       rdfs:subPropertyOf prov:qualifiedInfluence ;
368                       prov:unqualifiedForm prov:wasGeneratedBy ;
369                       rdfs:isDefinedBy ns:prov-o# .
370
371   ### http://www.w3.org/ns/prov#qualifiedInfluence
372   prov:qualifiedInfluence rdf:type owl:ObjectProperty ;
373                         rdfs:label "qualifiedInfluence" ;
374                         rdfs:comment "Because prov:qualifiedInfluence is a broad relation, the more specific relations (
                                qualifiedCommunication, qualifiedDelegation, qualifiedEnd, etc.) should be used when applicable.
                                "@en ;
375                         prov:category "qualified" ;
376                         prov:inverse "qualifiedInfluenceOf" ;
377                         prov:component "derivations" ;
378                         rdfs:range prov:Influence ;
379                         prov:sharesDefinitionWith prov:Influence ;
380                         prov:unqualifiedForm prov:wasInfluencedBy ;
381                         rdfs:isDefinedBy ns:prov-o# ;
382                         rdfs:domain [ rdf:type owl:Class ;
383                                     owl:unionOf ( prov:Activity
384                                                   prov:Agent
385                                                   prov:Entity
386                                                 )
387                                   ] .
388
389   ### http://www.w3.org/ns/prov#qualifiedUsage
390   prov:qualifiedUsage rdf:type owl:ObjectProperty ;
391                       rdfs:label "qualifiedUsage" ;
392                       prov:category "qualified" ;
393                       prov:inverse "qualifiedUsingActivity" ;
394                       prov:component "entities-activities" ;
395                       rdfs:comment "If this Activity prov:used Entity :e, then it can qualify how it used it using prov:
                              qualifiedUsage [ a prov:Usage; prov:entity :e; :foo :bar ]."@en ;
396                       rdfs:domain prov:Activity ;
397                       prov:sharesDefinitionWith prov:Usage ;
398                       rdfs:range prov:Usage ;
399                       rdfs:subPropertyOf prov:qualifiedInfluence ;
400                       prov:unqualifiedForm prov:used ;
401                       rdfs:isDefinedBy ns:prov-o# .
402
403   ### http://www.w3.org/ns/prov#specializationOf
404   prov:specializationOf rdf:type owl:ObjectProperty ;
405                         rdfs:label "specializationOf" ;
406                         prov:constraints "http://www.w3.org/TR/2012/WD-prov-dm-20120703/prov-constraints.html#prov-dm-
                                constraints-fig"^^xsd:anyURI ;
407                         prov:dm "http://www.w3.org/TR/2012/WD-prov-dm-20120703/prov-dm.html#term-specialization"^^xsd:anyURI ;
408                         prov:n "http://www.w3.org/TR/2012/WD-prov-dm-20120703/prov-n.html#expression-specialization"^^xsd:anyURI
                                ;
409                         prov:constraints "http://www.w3.org/TR/2013/REC-prov-constraints-20130430/#prov-dm-constraints-fig"^^xsd
                                :anyURI ;
410                         prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-specialization"^^xsd:anyURI ;
411                         prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-specialization"^^xsd:anyURI ;
412                         prov:component "alternate" ;
413                         prov:category "expanded" ;
414                         prov:inverse "generalizationOf" ;
415                         prov:definition "An entity that is a specialization of another shares all aspects of the latter, and
```

```
                        additionally presents more specific aspects of the same thing as the latter. In particular, the
                        lifetime of the entity being specialized contains that of any specialization. Examples of aspects
                         include a time period, an abstraction, and a context associated with the entity."@en ;
416                   rdfs:subPropertyOf owl:topObjectProperty ;
417                   rdfs:isDefinedBy prov: ;
418                   rdfs:domain prov:Entity ;
419                   rdfs:range prov:Entity ;
420                   rdfs:seeAlso prov:alternateOf ;
421                   rdfs:subPropertyOf prov:alternateOf ;
422                   rdfs:isDefinedBy ns:prov-o# .
423
424   ### http://www.w3.org/ns/prov#used
425   prov:used rdf:type owl:ObjectProperty ;
426           rdfs:label "used" ;
427           prov:inverse "wasUsedBy" ;
428           rdfs:comment "A prov:Entity that was used by this prov:Activity. For example, :baking prov:used :spoon, :egg, :
                  oven ."@en ;
429           prov:category "starting-point" ;
430           prov:component "entities-activities" ;
431           rdfs:domain prov:Activity ;
432           rdfs:range prov:Entity ;
433           prov:qualifiedForm prov:Usage ,
434                              prov:qualifiedUsage ;
435           rdfs:subPropertyOf prov:wasInfluencedBy ;
436           rdfs:isDefinedBy ns:prov-o# ;
437           owl:propertyChainAxiom ( prov:qualifiedUsage
438                                    prov:entity
439                                  ) .
440
441   ### http://www.w3.org/ns/prov#wasAssociatedWith
442   prov:wasAssociatedWith rdf:type owl:ObjectProperty ;
443                       rdfs:label "wasAssociatedWith" ;
444                       prov:component "agents-responsibility" ;
445                       prov:inverse "wasAssociateFor" ;
446                       rdfs:comment "An prov:Agent that had some (unspecified) responsibility for the occurrence of this prov:
                              Activity."@en ;
447                       prov:category "starting-point" ;
448                       rdfs:domain prov:Activity ;
449                       rdfs:range prov:Agent ;
450                       prov:qualifiedForm prov:Association ,
451                                          prov:qualifiedAssociation ;
452                       rdfs:subPropertyOf prov:wasInfluencedBy ;
453                       rdfs:isDefinedBy ns:prov-o# ;
454                       owl:propertyChainAxiom ( prov:qualifiedAssociation
455                                                prov:agent
456                                              ) .
457
458   ### http://www.w3.org/ns/prov#wasDerivedFrom
459   prov:wasDerivedFrom rdf:type owl:ObjectProperty ;
460                     rdfs:label "wasDerivedFrom" ;
461                     prov:inverse "hadDerivation" ;
462                     prov:definition "A derivation is a transformation of an entity into another, an update of an entity
                            resulting in a new one, or the construction of a new entity based on a pre-existing entity."@en ;
463                     prov:category "starting-point" ;
464                     rdfs:comment "The more specific subproperties of prov:wasDerivedFrom (i.e., prov:wasQuotedFrom, prov:
```

```
                        wasRevisionOf, prov:hadPrimarySource) should be used when applicable."@en ;
465                 prov:component "derivations" ;
466                 prov:qualifiedForm prov:Derivation ;
467                 rdfs:range prov:Entity ;
468                 rdfs:domain prov:Entity ;
469                 prov:qualifiedForm prov:qualifiedDerivation ;
470                 rdfs:subPropertyOf prov:wasInfluencedBy ;
471                 rdfs:isDefinedBy ns:prov-o# ;
472                 owl:propertyChainAxiom ( prov:qualifiedDerivation
473                                          prov:entity
474                                        ) .
475 [ rdf:type owl:Axiom ;
476   rdfs:comment "Derivation is a particular case of trace (see http://www.w3.org/TR/prov-dm/#term-trace), since it links an
          entity to another entity that contributed to its existence." ;
477   owl:annotatedProperty rdfs:subPropertyOf ;
478   owl:annotatedSource prov:wasDerivedFrom ;
479   owl:annotatedTarget prov:wasInfluencedBy
480 ] .
481
482 ### http://www.w3.org/ns/prov#wasGeneratedBy
483 prov:wasGeneratedBy rdf:type owl:ObjectProperty ;
484                 rdfs:label "wasGeneratedBy" ;
485                 prov:inverse "generated" ;
486                 prov:category "starting-point" ;
487                 prov:component "entities-activities" ;
488                 rdfs:isDefinedBy prov: ;
489                 rdfs:range prov:Activity ;
490                 rdfs:domain prov:Entity ;
491                 prov:qualifiedForm prov:Generation ;
492                 owl:inverseOf prov:generated ;
493                 prov:qualifiedForm prov:qualifiedGeneration ;
494                 rdfs:subPropertyOf prov:wasInfluencedBy ;
495                 rdfs:isDefinedBy ns:prov-o# ;
496                 owl:propertyChainAxiom ( prov:qualifiedGeneration
497                                          prov:activity
498                                        ) .
499
500 ### http://www.w3.org/ns/prov#wasInfluencedBy
501 prov:wasInfluencedBy rdf:type owl:ObjectProperty ;
502                  rdfs:label "wasInfluencedBy" ;
503                  rdfs:comment "Because prov:wasInfluencedBy is a broad relation, its more specific subproperties (e.g.
                        prov:wasInformedBy, prov:actedOnBehalfOf, prov:wasEndedBy, etc.) should be used when applicable."@
                        en ;
504                  prov:editorialNote """The sub-properties of prov:wasInfluencedBy can be elaborated in more detail using
                        the Qualification Pattern. For example, the binary relation :baking prov:used :spoon can be
                        qualified by asserting :baking prov:qualifiedUsage [ a prov:Usage; prov:entity :spoon; prov:
                        atLocation :kitchen ] .
505 Subproperties of prov:wasInfluencedBy may also be asserted directly without being qualified.
506 prov:wasInfluencedBy should not be used without also using one of its subproperties.
507 """@en ;
508                  rdfs:comment "This property has multiple RDFS domains to suit multiple OWL Profiles. See <a href=\"#owl-
                        profile\">PROV-O OWL Profile</a>." ;
509                  prov:category "qualified" ;
510                  prov:inverse "influenced" ;
511                  prov:component "agents-responsibility" ;
```

```
                    rdfs:isDefinedBy prov: ;
                    prov:qualifiedForm prov:Influence ;
                    prov:sharesDefinitionWith prov:Influence ;
                    owl:inverseOf prov:influenced ;
                    prov:qualifiedForm prov:qualifiedInfluence ;
                    rdfs:isDefinedBy ns:prov-o# ;
                    rdfs:domain [ rdf:type owl:Class ;
                                  owl:unionOf ( prov:Activity
                                                prov:Agent
                                                prov:Entity
                                              )
                                ] ;
                    rdfs:range [ rdf:type owl:Class ;
                                 owl:unionOf ( prov:Activity
                                               prov:Agent
                                               prov:Entity
                                             )
                               ] .
[ rdf:type owl:Axiom ;
  prov:definition "influencer: an identifier (o1) for an ancestor entity, activity, or agent that the former depends on;" ;
  owl:annotatedProperty rdfs:range ;
  owl:annotatedSource prov:wasInfluencedBy ;
  owl:annotatedTarget [ rdf:type owl:Class ;
                        owl:unionOf ( prov:Activity
                                      prov:Agent
                                      prov:Entity
                                    )
                      ]
] .
[ rdf:type owl:Axiom ;
  prov:definition "influencee: an identifier (o2) for an entity, activity, or agent; " ;
  owl:annotatedProperty rdfs:domain ;
  owl:annotatedSource prov:wasInfluencedBy ;
  owl:annotatedTarget [ rdf:type owl:Class ;
                        owl:unionOf ( prov:Activity
                                      prov:Agent
                                      prov:Entity
                                    )
                      ]
] .

###############################
# Classes
###############################
### http://ns.inria.fr/ratio4ta/v3#Data
:Data rdf:type owl:Class ;
      rdfs:label "Data"@en ;
      rdfs:subClassOf rdfg:Graph ,
                      prov:Entity ;
      prov:definition "A data is a set of RDF statements."@en .

### http://ns.inria.fr/ratio4ta/v3#DataDerivation
:DataDerivation rdf:type owl:Class ;
                rdfs:label "DataDerivation"@en ;
                rdfs:subClassOf prov:Association ,
```

```
                            prov:Derivation ;
                prov:definition "A data derivation represents a derivation that is performed as part of a reasoning process."
                      @en .

### http://ns.inria.fr/ratio4ta/v3#ExplanationBundle
:ExplanationBundle rdf:type owl:Class ;
                   rdfs:label "ExplanationBundle"@en ;
                   rdfs:subClassOf rdfg:Graph ,
                                   prov:Bundle ;
                   prov:definition "An explanation bundle is a set of RDF statements which explain how a data was derived."@
                         en .

### http://ns.inria.fr/ratio4ta/v3#InputData
:InputData rdf:type owl:Class ;
           rdfs:label "InputData"@en ;
           rdfs:subClassOf :Data ;
           prov:definition "An input data represents an input data (a set of RDF statements) used by a reasoning process."@en
                  .

### http://ns.inria.fr/ratio4ta/v3#OutputData
:OutputData rdf:type owl:Class ;
            rdfs:label "OutputData"@en ;
            rdfs:subClassOf :Data ;
            prov:definition "An output data represents an output data by a reasoning process."@en .

### http://ns.inria.fr/ratio4ta/v3#ReasoningProcess
:ReasoningProcess rdf:type owl:Class ;
                  rdfs:label "ReasoningProcess"@en ;
                  rdfs:subClassOf prov:Activity ;
                  prov:definition "A reasoning process represents a reasoning process of a software application. A reasoning
                        process uses input data and computes results. Each of these computed results includes output data.
                        Data derivations may be performed as part of a reasoning process which may lead to producing new data
                         that were not explicitly given in the input data."@en .

### http://ns.inria.fr/ratio4ta/v3#Result
:Result rdf:type owl:Class ;
        rdfs:label "Result"@en ;
        rdfs:subClassOf prov:Association ,
                        prov:Generation ;
        prov:definition "A result represents a result computed by a reasoning proces. "@en .

### http://ns.inria.fr/ratio4ta/v3#Rule
:Rule rdf:type owl:Class ;
      rdfs:label "Rule"@en ;
      rdfs:subClassOf prov:Plan ;
      prov:definition "A rule represents a rule that a reasoning process uses for a data derivation."@en .

### http://ns.inria.fr/ratio4ta/v3#SoftwareApplication
:SoftwareApplication rdf:type owl:Class ;
                     rdfs:label "SoftwareApplication"@en ;
                     rdfs:subClassOf prov:SoftwareAgent ;
                     prov:definition "A software application consumes and produces data."@en .

### http://www.w3.org/2004/03/trix/rdfg-1/Graph
rdfg:Graph rdf:type owl:Class ;
```

```
616           rdfs:label "Graph" ;
617           rdfs:comment "An RDF graph (with intensional semantics)." .
618
619   ### http://www.w3.org/ns/prov#Activity
620   prov:Activity rdf:type owl:Class ;
621               rdfs:label "Activity" ;
622               owl:disjointWith prov:Entity ;
623               prov:constraints "http://www.w3.org/TR/2013/REC-prov-constraints-20130430/#prov-dm-constraints-fig"^^xsd:anyURI
                        ;
624               prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-Activity"^^xsd:anyURI ;
625               prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-Activity"^^xsd:anyURI ;
626               prov:component "entities-activities" ;
627               prov:category "starting-point" ;
628               prov:definition "An activity is something that occurs over a period of time and acts upon or with entities; it
                      may include consuming, processing, transforming, modifying, relocating, using, or generating entities." ;
629               rdfs:isDefinedBy ns:prov-o# .
630
631   ### http://www.w3.org/ns/prov#ActivityInfluence
632   prov:ActivityInfluence rdf:type owl:Class ;
633                       rdfs:label "ActivityInfluence" ;
634                       rdfs:subClassOf prov:Influence ,
635                                       [ rdf:type owl:Restriction ;
636                                         owl:onProperty prov:hadActivity ;
637                                         owl:maxCardinality "0"^^xsd:nonNegativeInteger
638                                       ] ;
639                       owl:disjointWith prov:EntityInfluence ;
640                       prov:editorsDefinition "ActivitiyInfluence is the capacity of an activity to have an effect on the
                              character, development, or behavior of another by means of generation, invalidation,
                              communication, or other."@en ;
641                       rdfs:comment "ActivityInfluence provides additional descriptions of an Activity's binary influence upon
                               any other kind of resource. Instances of ActivityInfluence use the prov:activity property to
                              cite the influencing Activity."@en ,
642                                    "It is not recommended that the type ActivityInfluence be asserted without also asserting
                                          one of its more specific subclasses."@en ;
643                       prov:category "qualified" ;
644                       rdfs:seeAlso prov:activity ;
645                       rdfs:isDefinedBy ns:prov-o# .
646
647   ### http://www.w3.org/ns/prov#Agent
648   prov:Agent rdf:type owl:Class ;
649             rdfs:label "Agent" ;
650             owl:disjointWith prov:InstantaneousEvent ;
651             prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-agent"^^xsd:anyURI ;
652             prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-Agent"^^xsd:anyURI ;
653             prov:definition "An agent is something that bears some form of responsibility for an activity taking place, for
                    the existence of an entity, or for another agent's activity. "@en ;
654             prov:category "starting-point" ;
655             prov:component "agents-responsibility" ;
656             rdfs:isDefinedBy ns:prov-o# .
657
658   ### http://www.w3.org/ns/prov#AgentInfluence
659   prov:AgentInfluence rdf:type owl:Class ;
660                     rdfs:label "AgentInfluence" ;
661                     rdfs:subClassOf prov:Influence ;
662                     prov:editorsDefinition "AgentInfluence is the capacity of an agent to have an effect on the character,
```

```
                development, or behavior of another by means of attribution, association, delegation, or other."@en
                 ;
663             rdfs:comment "AgentInfluence provides additional descriptions of an Agent's binary influence upon any
                      other kind of resource. Instances of AgentInfluence use the prov:agent property to cite the
                      influencing Agent."@en ;
664             prov:category "qualified" ;
665             rdfs:comment "It is not recommended that the type AgentInfluence be asserted without also asserting one
                      of its more specific subclasses."@en ;
666             rdfs:seeAlso prov:agent ;
667             rdfs:isDefinedBy ns:prov-o# .
668
669 ### http://www.w3.org/ns/prov#Association
670 prov:Association rdf:type owl:Class ;
671             rdfs:label "Association" ;
672             rdfs:subClassOf prov:AgentInfluence ;
673             prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-Association"^^xsd:anyURI ;
674             prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-Association"^^xsd:anyURI ;
675             prov:component "agents-responsibility" ;
676             rdfs:comment "An instance of prov:Association provides additional descriptions about the binary prov:
                      wasAssociatedWith relation from an prov:Activity to some prov:Agent that had some responsiblity for it
                      . For example, :baking prov:wasAssociatedWith :baker; prov:qualifiedAssociation [ a prov:Association;
                      prov:agent :baker; :foo :bar ]."@en ;
677             prov:category "qualified" ;
678             prov:definition "An activity association is an assignment of responsibility to an agent for an activity,
                      indicating that the agent had a role in the activity. It further allows for a plan to be specified,
                      which is the plan intended by the agent to achieve some goals in the context of this activity."@en ;
679             prov:unqualifiedForm prov:wasAssociatedWith ;
680             rdfs:isDefinedBy ns:prov-o# .
681
682 ### http://www.w3.org/ns/prov#Bundle
683 prov:Bundle rdf:type owl:Class ;
684           rdfs:label "Bundle" ;
685           rdfs:subClassOf prov:Entity ;
686           prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-bundle-entity"^^xsd:anyURI ;
687           prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-bundle-declaration"^^xsd:anyURI ;
688           prov:category "expanded" ;
689           prov:definition "A bundle is a named set of provenance descriptions, and is itself an Entity, so allowing
                    provenance of provenance to be expressed."@en ;
690           rdfs:comment "Note that there are kinds of bundles (e.g. handwritten letters, audio recordings, etc.) that are
                    not expressed in PROV-O, but can be still be described by PROV-O."@en ;
691           rdfs:isDefinedBy ns:prov-o# .
692
693 ### http://www.w3.org/ns/prov#Delegation
694 prov:Delegation rdf:type owl:Class ;
695             rdfs:label "Delegation" ;
696             rdfs:subClassOf prov:AgentInfluence ;
697             prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-delegation"^^xsd:anyURI ;
698             prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-delegation"^^xsd:anyURI ;
699             prov:category "qualified" ;
700             rdfs:comment "An instance of prov:Delegation provides additional descriptions about the binary prov:
                    actedOnBehalfOf relation from a performing prov:Agent to some prov:Agent for whom it was performed. For
                     example, :mixing prov:wasAssociatedWith :toddler . :toddler prov:actedOnBehalfOf :mother; prov:
                    qualifiedDelegation [ a prov:Delegation; prov:entity :mother; :foo :bar ]."@en ;
701             prov:definition """Delegation is the assignment of authority and responsibility to an agent (by itself or by
                    another agent) to carry out a specific activity as a delegate or representative, while the agent it
```

```
                    acts on behalf of retains some responsibility for the outcome of the delegated work.
For example, a student acted on behalf of his supervisor, who acted on behalf of the department chair, who acted on behalf
      of the university; all those agents are responsible in some way for the activity that took place but we do not say
      explicitly who bears responsibility and to what degree."""@en ;
              prov:component "agents-responsibility" ;
              prov:unqualifiedForm prov:actedOnBehalfOf ;
              rdfs:isDefinedBy ns:prov-o# .

### http://www.w3.org/ns/prov#Derivation
prov:Derivation rdf:type owl:Class ;
              rdfs:label "Derivation" ;
              rdfs:subClassOf prov:EntityInfluence ;
              prov:constraints "http://www.w3.org/TR/2013/REC-prov-constraints-20130430/#prov-dm-constraints-fig"^^xsd:
                    anyURI ;
              prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-Derivation"^^xsd:anyURI ;
              prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#Derivation-Relation"^^xsd:anyURI ;
              prov:definition "A derivation is a transformation of an entity into another, an update of an entity resulting
                     in a new one, or the construction of a new entity based on a pre-existing entity."@en ;
              prov:component "derivations" ;
              rdfs:comment "An instance of prov:Derivation provides additional descriptions about the binary prov:
                    wasDerivedFrom relation from some derived prov:Entity to another prov:Entity from which it was derived.
                     For example, :chewed_bubble_gum prov:wasDerivedFrom :unwrapped_bubble_gum; prov:qualifiedDerivation [
                    a prov:Derivation; prov:entity :unwrapped_bubble_gum; :foo :bar ]."@en ,
                         "The more specific forms of prov:Derivation (i.e., prov:Revision, prov:Quotation, prov:
                               PrimarySource) should be asserted if they apply."@en ;
              prov:category "qualified" ;
              prov:unqualifiedForm prov:wasDerivedFrom ;
              rdfs:isDefinedBy ns:prov-o# .

### http://www.w3.org/ns/prov#End
prov:End rdf:type owl:Class ;
        rdfs:label "End" ;
        rdfs:subClassOf prov:EntityInfluence ,
                        prov:InstantaneousEvent ;
        prov:constraints "http://www.w3.org/TR/2013/REC-prov-constraints-20130430/#prov-dm-constraints-fig"^^xsd:anyURI ;
        prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-End"^^xsd:anyURI ;
        prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-End"^^xsd:anyURI ;
        rdfs:comment "An instance of prov:End provides additional descriptions about the binary prov:wasEndedBy relation
              from some ended prov:Activity to an prov:Entity that ended it. For example, :ball_game prov:wasEndedBy :
              buzzer; prov:qualifiedEnd [ a prov:End; prov:entity :buzzer; :foo :bar; prov:atTime '2012-03-09T08
              :05:08-05:00'^^xsd:dateTime ]."@en ;
        prov:category "qualified" ;
        prov:definition "End is when an activity is deemed to have been ended by an entity, known as trigger. The activity
              no longer exists after its end. Any usage, generation, or invalidation involving an activity precedes the
              activity's end. An end may refer to a trigger entity that terminated the activity, or to an activity, known
              as ender that generated the trigger."@en ;
        prov:component "entities-activities" ;
        prov:unqualifiedForm prov:wasEndedBy ;
        rdfs:isDefinedBy ns:prov-o# .

### http://www.w3.org/ns/prov#Entity
prov:Entity rdf:type owl:Class ;
           rdfs:label "Entity" ;
           owl:disjointWith prov:InstantaneousEvent ;
           prov:constraints "http://www.w3.org/TR/2013/REC-prov-constraints-20130430/#prov-dm-constraints-fig"^^xsd:anyURI ;
```

```
742           prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-entity"^^xsd:anyURI ;
743           prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-Entity"^^xsd:anyURI ;
744           prov:component "entities-activities" ;
745           prov:definition "An entity is a physical, digital, conceptual, or other kind of thing with some fixed aspects;
                  entities may be real or imaginary. "@en ;
746           prov:category "starting-point" ;
747           rdfs:isDefinedBy ns:prov-o# .
748
749   ### http://www.w3.org/ns/prov#EntityInfluence
750   prov:EntityInfluence rdf:type owl:Class ;
751                    rdfs:label "EntityInfluence" ;
752                    rdfs:subClassOf prov:Influence ;
753                    prov:editorsDefinition "EntityInfluence is the capacity of an entity to have an effect on the character,
                           development, or behavior of another by means of usage, start, end, derivation, or other. "@en ;
754                    rdfs:comment "EntityInfluence provides additional descriptions of an Entity's binary influence upon any
                           other kind of resource. Instances of EntityInfluence use the prov:entity property to cite the
                           influencing Entity."@en ,
755                                "It is not recommended that the type EntityInfluence be asserted without also asserting one
                                      of its more specific subclasses."@en ;
756                    prov:category "qualified" ;
757                    rdfs:seeAlso prov:entity ;
758                    rdfs:isDefinedBy ns:prov-o# .
759
760   ### http://www.w3.org/ns/prov#Generation
761   prov:Generation rdf:type owl:Class ;
762                rdfs:label "Generation" ;
763                rdfs:subClassOf prov:ActivityInfluence ,
764                                prov:InstantaneousEvent ;
765                prov:constraints "http://www.w3.org/TR/2013/REC-prov-constraints-20130430/#prov-dm-constraints-fig"^^xsd:
                       anyURI ;
766                prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-Generation"^^xsd:anyURI ;
767                prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-Generation"^^xsd:anyURI ;
768                rdfs:comment "An instance of prov:Generation provides additional descriptions about the binary prov:
                       wasGeneratedBy relation from a generated prov:Entity to the prov:Activity that generated it. For
                       example, :cake prov:wasGeneratedBy :baking; prov:qualifiedGeneration [ a prov:Generation; prov:activity
                        :baking; :foo :bar ]."@en ;
769                prov:category "qualified" ;
770                prov:component "entities-activities" ;
771                prov:definition "Generation is the completion of production of a new entity by an activity. This entity did
                       not exist before generation and becomes available for usage after this generation."@en ;
772                prov:unqualifiedForm prov:wasGeneratedBy ;
773                rdfs:isDefinedBy ns:prov-o# .
774
775   ### http://www.w3.org/ns/prov#Influence
776   prov:Influence rdf:type owl:Class ;
777                rdfs:label "Influence" ;
778                prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-influence"^^xsd:anyURI ;
779                prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-influence"^^xsd:anyURI ;
780                prov:component "derivations" ;
781                rdfs:comment "An instance of prov:Influence provides additional descriptions about the binary prov:
                       wasInfluencedBy relation from some influenced Activity, Entity, or Agent to the influencing Activity,
                       Entity, or Agent. For example, :stomach_ache prov:wasInfluencedBy :spoon; prov:qualifiedInfluence [ a
                       prov:Influence; prov:entity :spoon; :foo :bar ] . Because prov:Influence is a broad relation, the more
                       specific relations (Communication, Delegation, End, etc.) should be used when applicable."@en ,
782                            "Because prov:Influence is a broad relation, its most specific subclasses (e.g. prov:Communication
```

```
                              , prov:Delegation, prov:End, prov:Revision, etc.) should be used when applicable."@en ;
783             prov:category "qualified" ;
784             prov:definition "Influence is the capacity of an entity, activity, or agent to have an effect on the character
                      , development, or behavior of another by means of usage, start, end, generation, invalidation,
                      communication, derivation, attribution, association, or delegation."@en ;
785             prov:unqualifiedForm prov:wasInfluencedBy ;
786             rdfs:isDefinedBy ns:prov-o# .
787
788 ### http://www.w3.org/ns/prov#InstantaneousEvent
789 prov:InstantaneousEvent rdf:type owl:Class ;
790                     rdfs:label "InstantaneousEvent" ;
791                     prov:constraints "http://www.w3.org/TR/2013/REC-prov-constraints-20130430/#dfn-event"^^xsd:anyURI ;
792                     prov:component "entities-activities" ;
793                     rdfs:comment "An instantaneous event, or event for short, happens in the world and marks a change in
                            the world, in its activities and in its entities. The term 'event' is commonly used in process
                            algebra with a similar meaning. Events represent communications or interactions; they are
                            assumed to be atomic and instantaneous."@en ;
794                     prov:definition "The PROV data model is implicitly based on a notion of instantaneous events (or just
                            events), that mark transitions in the world. Events include generation, usage, or invalidation
                            of entities, as well as starting or ending of activities. This notion of event is not first-
                            class in the data model, but it is useful for explaining its other concepts and its semantics."@
                            en ;
795                     prov:category "qualified" ;
796                     rdfs:isDefinedBy ns:prov-o# .
797
798 ### http://www.w3.org/ns/prov#Invalidation
799 prov:Invalidation rdf:type owl:Class ;
800                 rdfs:label "Invalidation" ;
801                 rdfs:subClassOf prov:ActivityInfluence ,
802                                 prov:InstantaneousEvent ;
803                 prov:constraints "http://www.w3.org/TR/2013/REC-prov-constraints-20130430/#prov-dm-constraints-fig"^^xsd:
                          anyURI ;
804                 prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-Invalidation"^^xsd:anyURI ;
805                 prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-Invalidation"^^xsd:anyURI ;
806                 prov:component "entities-activities" ;
807                 prov:definition "Invalidation is the start of the destruction, cessation, or expiry of an existing entity
                          by an activity. The entity is no longer available for use (or further invalidation) after
                          invalidation. Any generation or usage of an entity precedes its invalidation." ;
808                 prov:category "qualified" ;
809                 rdfs:comment "An instance of prov:Invalidation provides additional descriptions about the binary prov:
                          wasInvalidatedBy relation from an invalidated prov:Entity to the prov:Activity that invalidated it.
                          For example, :uncracked_egg prov:wasInvalidatedBy :baking; prov:qualifiedInvalidation [ a prov:
                          Invalidation; prov:activity :baking; :foo :bar ]."@en ;
810                 prov:unqualifiedForm prov:wasInvalidatedBy ;
811                 rdfs:isDefinedBy ns:prov-o# .
812
813 ### http://www.w3.org/ns/prov#Plan
814 prov:Plan rdf:type owl:Class ;
815           rdfs:label "Plan" ;
816           rdfs:subClassOf prov:Entity ;
817           prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-Association"^^xsd:anyURI ;
818           prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-Association"^^xsd:anyURI ;
819           prov:definition "A plan is an entity that represents a set of actions or steps intended by one or more agents to
                  achieve some goals." ;
820           prov:category "expanded" ;
```

```
821         rdfs:comment "There exist no prescriptive requirement on the nature of plans, their representation, the actions or
                steps they consist of, or their intended goals. Since plans may evolve over time, it may become necessary
               to track their provenance, so plans themselves are entities. Representing the plan explicitly in the
               provenance can be useful for various tasks: for example, to validate the execution as represented in the
               provenance record, to manage expectation failures, or to provide explanations."@en ;
822         prov:category "qualified" ;
823         prov:component "agents-responsibility" ;
824         rdfs:isDefinedBy ns:prov-o# .
825
826 ### http://www.w3.org/ns/prov#SoftwareAgent
827 prov:SoftwareAgent rdf:type owl:Class ;
828                  rdfs:label "SoftwareAgent" ;
829                  rdfs:subClassOf owl:Thing ,
830                                  prov:Agent ;
831                  prov:dm "http://www.w3.org/TR/2012/WD-prov-dm-20120703/prov-dm.html#term-agent"^^xsd:anyURI ;
832                  prov:n "http://www.w3.org/TR/2012/WD-prov-dm-20120703/prov-n.html#expression-types"^^xsd:anyURI ;
833                  prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-agent"^^xsd:anyURI ;
834                  prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-types"^^xsd:anyURI ;
835                  prov:component "agents-responsibility" ;
836                  prov:definition "A software agent is running software."@en ;
837                  prov:category "expanded" ;
838                  rdfs:isDefinedBy prov: ,
839                                   ns:prov-o# .
840
841 ### http://www.w3.org/ns/prov#Start
842 prov:Start rdf:type owl:Class ;
843           rdfs:label "Start" ;
844           rdfs:subClassOf prov:EntityInfluence ,
845                           prov:InstantaneousEvent ;
846           prov:constraints "http://www.w3.org/TR/2013/REC-prov-constraints-20130430/#prov-dm-constraints-fig"^^xsd:anyURI ;
847           prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-Start"^^xsd:anyURI ;
848           prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-Start"^^xsd:anyURI ;
849           prov:component "entities-activities" ;
850           prov:category "qualified" ;
851           rdfs:comment "An instance of prov:Start provides additional descriptions about the binary prov:wasStartedBy
                 relation from some started prov:Activity to an prov:Entity that started it. For example, :foot_race prov:
                 wasStartedBy :bang; prov:qualifiedStart [ a prov:Start; prov:entity :bang; :foo :bar; prov:atTime
                 '2012-03-09T08:05:08-05:00'^^xsd:dateTime ] ."@en ;
852           prov:definition "Start is when an activity is deemed to have been started by an entity, known as trigger. The
                 activity did not exist before its start. Any usage, generation, or invalidation involving an activity
                 follows the activity's start. A start may refer to a trigger entity that set off the activity, or to an
                 activity, known as starter, that generated the trigger."@en ;
853           prov:unqualifiedForm prov:wasStartedBy ;
854           rdfs:isDefinedBy ns:prov-o# .
855
856 ### http://www.w3.org/ns/prov#Usage
857 prov:Usage rdf:type owl:Class ;
858           rdfs:label "Usage" ;
859           rdfs:subClassOf prov:EntityInfluence ,
860                           prov:InstantaneousEvent ;
861           prov:constraints "http://www.w3.org/TR/2013/REC-prov-constraints-20130430/#prov-dm-constraints-fig"^^xsd:anyURI ;
862           prov:dm "http://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-Usage"^^xsd:anyURI ;
863           prov:n "http://www.w3.org/TR/2013/REC-prov-n-20130430/#expression-Usage"^^xsd:anyURI ;
864           rdfs:comment "An instance of prov:Usage provides additional descriptions about the binary prov:used relation from
                 some prov:Activity to an prov:Entity that it used. For example, :keynote prov:used :podium; prov:
```

```
            qualifiedUsage [ a prov:Usage; prov:entity :podium; :foo :bar ]."@en ;
865         prov:definition "Usage is the beginning of utilizing an entity by an activity. Before usage, the activity had not
            begun to utilize this entity and could not have been affected by the entity."@en ;
866         prov:category "qualified" ;
867         prov:component "entities-activities" ;
868         prov:unqualifiedForm prov:used ;
869         rdfs:isDefinedBy ns:prov-o# .
```

Appendix B

Introduction, Résumé et Conclusion en Français

B.1 Introduction

Le Web évolue, partant d'un Web de documents pour aller vers un web de données. Grace à l'initiative Linking Open Data du W3C, dans les dernières années, nous avons assisté à une forte croissance de la publication de données liées. Ceci est le résultat d'efforts communautaires, d'organismes gouvernementaux, de sites de réseaux sociaux, de communautés scientifiques, etc. Les fournisseurs de données viennent donc de différents domaines et publient leurs données de façon interconnectée à l'aide du modèle de données RDF et de points d'accès SPARQL pour permettre l'interrogation de leurs données, ce qui permet de créer un graphe mondial de données. Cela présente un énorme potentiel pour l'intégration de données disparates et pour soutenir une nouvelle génération d'applications intelligentes. Mais l'intégration des données liées à l'aide d'interrogations distantes peut induire des charges de calcul importantes avec des requêtes demandant énormément de ressources. La gestion de ces charges de calcul est essentielle pour l'intégration efficace des données liées. À cette fin, la compréhension du comportement de la requête avant même son exécution peut aider des utilisateurs tels que les administrateurs de la base de connaissances ou des développeurs d'applications dans leurs tâches de gestion de la charge de travail, dans la configuration, l'organisation, l'inspection et l'optimisation. Dans un second temps, dans l'environnement ouvert du Web où

des données liées hétérogènes sont échangées, intégrées, et matérialisées dans des référentiels distribués accessibles à travers des points d'accès SPARQL, comprendre le résultat de la requête est essentiel pour en juger la valididté. Les explications des résultats d'une requête permettent cette compréhension en fournissant des informations telles que les triplets ayant contribué aux résultats, comment ces triplets ont été combinés et qui a fourni ces triplets. En outre, les applications peuvent consommer des données liées, dont certaines peuvent être obtenues en interrogeant d'autres applications et par raisonnement sur les données consommées pour produire des résultats qui peuvent eux-mêmes devenir de nouvelles données liées. Dans ce contexte, il est essentiel d'expliquer non seulement les raisonnements faits par les applications, mais aussi tout le cycle de vie des données consommées, pour aider les utilisateurs à comprendre comment les résultats et les nouvelles données liées ont été obtenus. Ce genre d'explications peut devenir très important lorsque les applications consomment une grande quantité de données ou les données consommées proviennent d'une longue chaîne de dérivations. Dans ce contexte, fournir des explications avec des détails sur toutes les dérivations peut submerger les utilisateurs avec trop d'informations. Ils voudront peut-être avoir la capacité de se concentrer sur des parties spécifiques d'une explication, filtrer les informations d'une explication, ou obtenir des explications courtes avec des informations importantes. Dans la section suivante, nous discutons chacun de ces problèmes identifiés en introduction et identifions les questions de recherche correspondantes.

B.1.1 Questions de Recherche

La question de recherche globale que nous abordons dans cette thèse est:

RQ. Comment aider les utilisateurs à comprendre le comportement d'une requête et les résultats obtenus dans le contexte de la consommation de données liées?

Nous décomposons cette question en plusieurs sous-questions. Tout d'abord, nous abordons le problème de la compréhension du comportement de la requête

dans le contexte des données liées. Pour aider la compréhension du comportement d'une requête, nous visons à fournir des prévisions de performance aux utilisateurs. Les utilisateurs tels que les administrateurs de la base de connaissances peuvent utiliser ces prévisions pour permettre une gestion efficace de la charge de travail et pour assurer une qualité de service (QoS) spécifique. La question de la recherche dans ce contexte est la suivante:

RQ1. Comment prédire des indicateurs de performance des requêtes sur des points d'accès SPARQL qui fournissent des services d'interrogation des données liées?

Deuxièmement, nous abordons le problème de la fourniture des explications pour aider les utilisateurs à comprendre les résultats obtenus après l'exécution. Cette meilleure compréhension peut conduire à une meilleure confiance dans le système qui produit le résultat. Il existe deux cas pour la compréhension des résultats dans le contexte de la consommation de données liées: les résultats de la résolution d'une requête SPARQL et les résultats produits par les applications.

Pour les résultats des requêtes SPARQL, le principal défi est de fournir des explications pour des requêtes SPARQL alors que les systèmes sont administrés et contrôlés par des tiers. Par conséquent, la réingénierie du modèle sous-jacent aux données, du langage de requête, ou du processeur de requêtes afin de les amener à générer des métadonnées d'explications au cours du traitement de la requête ne sont pas possibles dans ce scénario. En outre, nous étudions l'impact des explications des résultats dans le contexte de la consommation de données liés. Les questions de recherche concernant ces aspects sont les suivantes:

RQ2. Comment fournir des explications pour les résultats des requêtes SPARQL sur les ponts d'accès SPARQL qui fournissent des services d'interrogation des données liées?

RQ3. Quels sont les impacts de la génération des explications sur le calcul des résultats à une requête?

A partir des résultats générés par les applications, le principal défi est de fournir des installations permettant la publication et l'échange des explications compte tenu de l'architecture distribuée et décentralisée du Web. Les applications peuvent utiliser des données qui sont distribuées à travers le Web. Les données consommées dans ce cadre peuvent être aussi des données dérivées. Nous étudions comment fournir des explications dans un tel scénario - qui explique non seulement le raisonnement par les applications, mais aussi les dérivations de données consommées. De plus, fournir des explications détaillées peut submerger les utilisateurs avec trop d'informations - particulièrement les utilisateurs non-experts. Dans ce contexte, le défi est de résumer les explications à fournir et générer des explications courtes. Compte tenu de ces problèmes, les questions de recherche sont les suivants:

RQ4. Comment fournir des explications pour les résultats produits par les applications qui consomment des données liées?

RQ5. Comment résumer les explications pour les résultats produits par les applications qui consomment des données liées?

B.2 Résumé de la Thèse

Cette thèse contient 8 chapitres:

1. Le chapitre 1 présente le contexte de la thèse, les questions de recherche, et donne un aperçu des principales contributions de cette thèse. Les données liées présentent un énorme potentiel pour l'intégration de quantités massives de données disparates pour soutenir une nouvelle génération d'applications intelligentes. L'intégration des données liées à l'aide d'interrogations peut induire des charges de calcul importantes. La gestion de ces charges de travail est essentielle pour l'intégration efficace de ces données. à cette fin, la compréhension du comportement des requêtes avant leur exécution peut aider les utilisateurs tels que les administrateurs de la base de connaissances ou les

développeurs d'applications à des tâches de gestion de la charge de travail. En outre, la compréhension des résultats est essentielle pour juger leur validité. Nous identifions cinq questions de recherche dans le but d'aider les utilisateurs à comprendre le comportement de la requête et le résultat des dérivations.

2. Le chapitre 2 passe en revue les sujets nécessaires à la connaissance de fond de cette thèse et fournit un état d'art des domaines connexes. Nous commençons par discuter l'évolution du Web d'un Web de documents vers un Web de données. Nous présentons ensuite les notions de RDF, SPARQL, et des données liées sur le Web. Nous discutons les principes des données liées en mettant l'accent sur l'édition et la consommation des données liées. Nous passons ensuite en revue la littérature sur l'assistance aux utilisateurs dans l'interrogation. Le travail examiné vise à aider les utilisateurs à l'interrogation sur trois aspects: le raffinement d'une requête existante, la construction de la requête, et la compréhension du comportement de la requête. Das cette thèse nous nous concentrons uniquement sur la compréhension du comportement des requêtes. Nous visons à aider les utilisateurs dans le comportement de recherche de compréhension sur les données liées avant l'exécution de la requête. Nous visons à aider les utilisateurs dans des tâches telles que la gestion de la charge de travail pour répondre aux exigences de qualité de service spécifiques et ceci en fournissant les prédictions des mesures de performance de la requête. Le principal défi à cet égard est de prévoir des mesures de performance de requête avant exécution de la requête à partir des caractéristiques SPARQL de cette requête. Les techniques traditionnelles d'estimation des coûts de requêtes SPARQL sont basées sur les statistiques et sur les données sous-jacentes. Cependant, les statistiques sur les données sous-jacentes sont souvent absentes des données liées. Nous étudions donc comment prédire les indicateurs de performance d'interrogation sans l'aide des statistiques sur les données sous-jacentes. Nous passons ensuite en revue la littérature sur

l'assistance aux utilisateurs dans la compréhension des résultats. Nous étudions les contributions aidant les utilisateurs à comprendre les résultats en fournissant des explications. Ces explications peuvent inclure la manipulation de l'information étape par étape par les différents algorithmes, les arbres de preuve de dérivations, les justifications des inférences, et la provenance des données. Nous étudions comment fournir des explications pour les résultats des requêtes SPARQL dans le contexte de données liées. Ces explications de résultats de requêtes sont basées sur le résultat de requêtes de provenance. Les techniques actuelles sont basées sur des approches d'annotation. Ces approches nécessitent la refonte du modèle de la base de données, du langage de requête, et du moteur de traitement d'une requête pour calculer la provenance lors du traitement de la requête. Cependant, ce type d'approche n'est pas une option dans le scénario des données liées qui sont hébergées, servies, et contrôlées par des tiers. Nous étudions comment calculer la provenance d'une requête SPARQL sans une telle reconception. En outre, très peu a été fait dans les travaux antérieurs dans la littérature du Web sémantique pour évaluer la validité des hypothèses telles que des explications permettraient d'améliorer la compréhension et la confiance des utilisateurs. Nous étudions l'impact des explications des résultats de la requête sur des utilisateurs de données liées. La plupart des travaux antérieurs sur les explications pour le Web sémantique ne traitent pas d'explications dans un environnement distribué. Nous étudions comment fournir des explications pour le scénario de données liées. Dans ce contexte, le défi consiste à fournir des explications pour des données distribuées produites par les applications de données liées distribués à travers le Web. Enfin, très peu d'approches existantes font face au problème de la synthèse des explications. Nous étudions comment fournir des explications résumées, des explications courtes et la possibilité de filtrer les informations importantes dans les explications.

3. Le chapitre 3 présente une approche pour prédire les performances des requêtes SPARQL dans le but d'aider les utilisateurs (par exemple, les administrateurs de la base de connaissances ou les développeurs d'applications) dans les tâches liées à la gestion de la charge de travail. Les administrateurs de la base de connaissances peuvent utiliser des indicateurs de performance prévus pour gérer efficacement les charges de travail telles que la qualité spécifique de service (QoS) et vérifier que les objectifs sont atteints. Les architectes de systèmes peuvent utiliser la prédiction des performances de requête pour estimer les configurations de système pour soutenir un type spécifique de la charge de travail. Les développeurs d'applications peuvent utiliser la prédiction des performances de requête de choisir parmi les requêtes alternatives en fonction des exigences de performance. La génération actuelle des méthodes d'estimation des coûts de requêtes SPARQL est basée sur les statistiques de données et des heuristiques. Les approches fondées sur les statistiques présentent deux inconvénients majeurs dans le contexte de données liées. Tout d'abord, les statistiques (par exemple, histogrammes) sur les données font souvent absents des scénarios de Linked Data parce qu'ils sont coûteux à produire et à entretenir. Deuxièmement, en raison du modèle de données basé sur les graphes et la liberté de schéma des données RDF l'efficacité des statistiques pour l'estimation du coût de requêtes n'est pas claire. Les approches basées sur des heuristiques ne nécessitent généralement pas de connaissances de données statistiques sous-jacentes. Cependant, elles sont fondées sur des hypothèses fortes telles que l'examen des requêtes de certaines structures moins chères que d'autres. Ces hypothèses peuvent tenir pour certains ensembles de données RDF et peuvent ne pas tenir pour d'autres. Nous adoptons une approche plutôt pragmatique d'estimation de coût de la requête SPARQL. Nous apprenons les performances des requêtes SPARQL déjà exécutées. Des travaux récents dans la recherche de base de données montrent que des indicateurs de performance peuvent

être prédits avec précision, sans aucune connaissance des statistiques de données en appliquant des techniques d'apprentissage automatique sur les journaux de requêtes déjà exécutées. De même, nous appliquons des techniques d'apprentissage automatique pour apprendre des mesures de performances des requêtes SPARQL à partir de requêtes déjà exécutées. Nous considérons temps d'exécution comme la mesure de la performance des requêtes. Nous discutons la façon de modéliser les caractéristiques de la requête SPARQL comme vecteurs de caractéristiques pour les algorithmes d'apprentissage machine, tels que les k plus proches voisins (k-NN) et les machines à support de vecteurs (SVM). Nous présentons nos expériences avec les requêtes de données liées communes et discutons nos résultats. Nous montrons des prédictions en temps d'exécution des requêtes très précises à l'aide de k-NN et SVM.

4. Le chapitre 4 traite d'aider les utilisateurs à comprendre les résultats de la requête. Nous présentons une approche pour expliquer les résultats de la requête SPARQL. Au sein de la communauté du Web sémantique, des explications ont été étudiées pour les applications du Web sémantique et les inférences OWL. L'explication des résultats de requêtes SPARQL n'a pas été étudiée de façon indépendante par la communauté. Cependant, il ya eu plusieurs travaux sur le traçage de l'origine des résultats de la requête - par exemple, la provenance (pourquoi). Ces tentatives sont basées sur ce qui est connu comme l'approche d'annotation où le modèle sous-jacent de données, le langage de requête, et le moteur de traitement des requêtes sont réorganisés pour calculer la provenance au cours du traitement de la requête. Cela n'est pas souhaitable pour le scénario de Linked Data car la refonte du modèle sous-jacent de données, le langage de requête, ou le processeur de requêtes est souvent impossible du côté de l'interrogation. Nous proposons une approche sans annotation pour générer la provenance (pourquoi) des résultats de la requête SPARQL. Nous générons l'explication avec une requête suplémentaire extrayant la provenance

(pourquoi). Nous générons la provenance (pourquoi) des résultats de la requête SPARQL sans modifier le modèle de données RDF, le langage de requête, ou le processeur de requêtes. Notre approche est appropriée pour les scénarios où les clients d'interrogation sont nécessaires pour générer provenance du côté de l'interrogation et ne sont pas autorisés à modifier le processeur de requêtes ou le modèle de données sous-jacente - le scénario Linked Data. En outre, les métadonnées de provenance sont générées uniquement lorsque cela est nécessaire - communément appelée l'approche paresseux. Par conséquent, par défaut notre approche n'ajoute pas de temps d'exécution de requêtes supplémentaire ou de stockage des métadonnées de provenance. Nous montrons la faisabilité de notre approche pour les requêtes de données liées classiques. Enfin, nous présentons un prototype de processeur de requêtes fédérées générant des explications.

5. Le chapitre 5 présente une étude sur les utilisateurs pour évaluer l'impact des explications de résultats de la requête. Une grande partie des travaux antérieurs sur les explications dans la littérature du Web sémantique a mis l'accent sur la représentation et la production d'explications. Des explications sont fournies pour aider les utilisateurs à améliorer leur compréhension du processus des résultats découlant et la circulation de l'information impliquée dans le processus. La compréhension améliorée peut conduire à une meilleure acceptation par les utilisateurs, et donc une meilleure confiance sur les applications du Web sémantique. Ces valeurs d'explications n'ont cependant pas été évaluées dans la littérature du Web sémantique. Dans ce chapitre, nous présentons une étude sur les utilisateurs qui évalue l'impact des résultats de requêtes d'explications dans le scénario de traitement des requêtes fédérées de données liées. En particulier, nous étudions si en fournissant des explications pour les résultats des requêtes fédérées on peut améliorer la compréhension des utilisateurs du processus de résolution de la requête, et les aider à porter des

jugements de confiance sur les résultats. Notre étude sur les utilisateurs montre que nos explications de résultats de requêtes sont utiles pour les utilisateurs finaux à comprendre les dérivations de résultats et à porter des jugements de confiance sur les résultats.

6. Le chapitre 6 décrit notre approche pour expliquer les résultats obtenus par les applications qui consomment des données liées. Les applications peuvent consommer des données liées, dont certaines peuvent être obtenues par d'autres applications, et par le raisonnement sur les données consommées pour produire des résultats ou même produire plus de données liées. Dans ce scénario distribué des données liées, il est essentiel d'expliquer non seulement le raisonnement par les applications, mais aussi les dérivations des données consommées, pour aider les utilisateurs (tels que les ingénieurs de la connaissance ou les utilisateurs finaux des applications de données liées) à comprendre comment les résultats ou de nouvelles données liées ont été tirées. Une grande partie des travaux antérieurs sur les explications pour le Web sémantique ne traite pas d'explication dans un environnement distribué. Une approche existante propose une solution centralisée de registre sur la base de la publication des métadonnées explication de raisonneurs distribués. Nous proposons une solution décentralisée à ce problème. Nous publions les métadonnées d'explication comme des données liées que nous appelons des Explications Liées. Dans cette approche, nous ne sommes pas contraints de publier les métadonnées d'explication dans un endroit centralisé comme dans les approches précédentes. Pour générer des explications, nous joingons aux données les URI dereferenceables de leur métadonnées de provenance et qui peuvent en suite être traitées pour être présentées sous une forme compréhensible. Pour publier les explications des métadonnées connexes, nous présentons un vocabulaire pour décrire les métadonnées et les lignes directrices pour publier ces métadonnées comme des données liées. Contrairement aux explications dont nous avons parlé dans

les chapitres 4 et 5, dans ce chapitre, nous fournissons des explications pour les résultats produits par les applications de données liées à base de règles génériques. Cela signifie que nous fournissons des explications pour des résultats de dérivations montrant les triplets utilisés dans une dérivation. En outre, si ces triplets utilisés ont également été calculés, nous fournissons des explications pour eux.

7. Le chapitre 7 présente une approche pour résumer les explications et filtrer les informations dans une explication basée sur des critères de filtrage spécifiés par l'utilisateur. Bien que les explications détaillées de toutes les étapes de dérivation puissent être utiles pour les utilisateurs expérimentés, elles peuvent aussi submerger les utilisateurs non-experts avec trop d'informations. De plus, un utilisateur expert comme un ingénieur de la connaissance peut vouloir se concentrer sur une partie spécifique d'une explication détaillée. Un ingénieur de la connaissance peut aussi vouloir une courte explication pour avoir un aperçu du raisonnement. Dans le chapitre 6, nous avons discuté la façon de fournir des explications complètes et des preuves fondées sur les arbres de dérivation pour les résultats produits par les applications qui consomment des données liées. Nous fournissons à partir de là des explications résumées. Nous définissons cinq mesures de résumé: (i) la saillance des déclarations RDF, (ii) la similitude des déclarations RDF en ce qui concerne les critères de filtrage des utilisateurs, (iii) l'abstraction des déclarations RDF par rapport à l'arbre de preuve, (iv) le poids de la sous-arborescence dans l'arbre de preuve - poids d'un noeud dans l'arbre de preuve, (v) la cohérence des déclarations RDF par rapport à l'arbre de preuve. Nous évaluons différentes combinaisons de ces mesures. L'évaluation montre que notre approche produit classements de haute qualité pour résumer les déclarations de l'explication. Les explications sont résumées également très précise avec des valeurs F-pointage de 0,6 à 0,72 pour les petits résumés.

8. Le chapitre 8 conclut la thèse avec un résumé de nos contributions et décrit nos perspectives comme le fait la section suivante.

B.3 Conclusion et Perspectives

B.3.1 Résumé des Contributions

Dans cette thèse, nous visons à aider les utilisateurs dans la compréhension du traitement de requêtes et des résultats dans le contexte de la consommation de données liées. Nous avons contribué dans cinq domaines: la prévision de la performance de requêtes, la provenance de résultats de requête, les explications pour les données liées, la publication des explications et le résumé des explications.

Prédiction de performances des requêtes. Nous présentons une approche d'apprentissage automatique pour prévoir des mesures de performance de requêtes. Nous apprenons le temps d'exécution des requêtes à partir de requêtes déjà exécutées - sans l'aide des statistiques sur les données RDF sous-jacents. Nous discutons la façon de modéliser les requêtes SPARQL comme des vecteurs de caractéristiques, et montrons des prédictions très précises. Les prévisions des mesures de performance de requête à l'aide de notre approche peuvent être utilisées pour aider les utilisateurs à comprendre les performances des requêtes pour des tâches liées à la gestion de la charge de travail pour atteindre les objectifs de qualité de service spécifiques dans le cadre de l'interrogation de données liées.

Résultat de requête et provenance. Nous présentons une approche sans annotation pour générer la provenance des résultats de la requête SPARQL et montrons la faisabilité pour les requêtes de données liées classiques. Nous présentons un prototype de processeur de requêtes fédérées générant de telles explications et détaillons les présentations de nos explications.

Évaluer les explications. Nous présentons une étude utilisateur pour évaluer l'impact des explications dans un scénario fédéré de traitement de requête pour les données liées. Notre étude sur les utilisateurs montre que nos explications de résultats de requête sont utiles pour aider les utilisateurs finaux à comprendre les dérivations de résultats et porter des jugements de confiance sur les résultats.

Explications pour les données liées. Nous discutons la façon de représenter et de générer des explications de données liées. Nous présentons le vocabulaire Ratio4TA pour décrire les métadonnées de l'explication et introduire la notion des Explications liées. Ceci permet d'expliquer des données distribuées de façon décentralisée. Ratio4TA étend l'ontologie standard PROV du W3C pour permettre aux consommateurs de données de traiter les métadonnées de l'explication selon les normes W3C PROV. Nous montrons aussi comment générer des explications en langage naturel à partir de ces métadonnées d'explication.

Résumer les explications pour les données liées. Bien que les explications avec les détails de toutes les étapes de dérivation puissent être utiles pour les utilisateurs expérimentés, elles peuvent submerger les utilisateurs non-experts avec trop d'informations. Nous avons présenté cinq mesures pour résumer les explications. Nous évaluons différentes combinaisons de ces mesures. L'évaluation montre que notre approche produit des classements de haute qualité pour résumer les déclarations de l'explication. Nos explications résumées sont très précises avec les valeurs F-scores allant de 0,6 à 0,72 pour de petits résumés.

B.3.2 Perspectives

Nous avons plusieurs perspectives pour notre prédiction de la performance des requêtes, pour l'explication de résultats de requêtes, et les explications liées.

B.3.2.1 Prédiction de performances des requêtes

à l'avenir, d'une part, nous aimerions utiliser notre approche dans l'optimisation des requêtes et la comparer aux techniques traditionnelles d'estimation de coût de requêtes dans des scénarios de Linked Data - par exemple l'optimisation de l'ordre dans le traitement de requêtes fédéré. Nous tenons à comparer notre approche à de telles approches. Deuxièmement, nous prévoyons de générer systématiquement les requêtes d'apprentissage pour deux scénarios: (a) les journaux de requêtes requêtes réelles (b) à partir d'un petit ensemble de requêtes échantillon. Nous prévoyons d'appliquer des techniques d'extraction de log de requêtes pour générer systématiquement les requêtes d'apprentissage. Des travaux récents sur la fouille de log de requêtes montrent que la majorité des requêtes SPARQL partagent certaines caractéristiques communes. Nous prévoyons de tenir compte de ces caractéristiques significatives statistiquement communes dans les requêtes d'apprentissage. Nous souhaitons également explorer comment ces caractéristiques communes peuvent être utilisées comme éléments descriptifs de la requête. Troisièmement, nous aimerions explorer des techniques d'apprentissage automatique en ligne pour nos modèles. Notre objectif serait d'affiner nos modèles de prévision basé sur les nouvelles prévisions et de leurs valeurs réelles. Enfin, nous aimerions inclure des caractéristiques de charge et de disponibilité des services d'interrogation. En ce sens, nous avons l'intention d'exécuter les requêtes de formation toutes les heures et inclure des charactéristiques telles que le temps, le jour et le mois. Cela nous aidera à modéliser les modèles de charge de travail pour les services SPARQL publics.

B.3.2.2 Explication de Résultats

Dans les travaux futurs, nous tenons à étendre notre algorithme pour générer la provenance (comment) qui explique comment un tuple résultat a été obtenu avec les détails des opérations effectuées dans le calcul. Les opérations SPARQL effectuées peuvent être extraites des modèles de requêtes de la même manière que nous

extrayons les triplets de la provenance (pourquoi). En fait, l'arbre d'expression algébrique que nous générons au cours du processus d'extraction de la provenance (pourquoi) contient déjà ces opérations. Pour la provenance (comment), il faudrait associer ces opérateurs aux triplets de la provenance (pourquoi) qu'ils concernent. Actuellement, nous présentons la première dérivation dans notre interface utilisateur d'explication. Il serait intéressant d'explorer comment nous pouvons présenter efficacement l'information de la provenance aux utilisateurs. En ce sens, une approche pourrait consister à classer les dérivations de la provenance, ce qui nous obligerait à définir des critères de classement pour les dérivations de la provenance. Enfin, notre étude sur les utilisateurs pour évaluer l'impact des résultats de requête explications ne comptait que 11 participants. Les participants devaient avoir quelques notions de RDF et SPARQL, et être motivés pour simuler un processus simple de résolution de requêtes fédérées. Il était difficile de trouver un grand nombre de participants. De plus, les participants ont été rendus anonymes et nous ne pouvions pas revenir vers les participants pour demander pourquoi un participant donné a fourni une réponse donnée. Une étude contrÃťlée par l'utilisateur avec un plus grand nombre de participants nous donnerait des résultats plus concluants et permettrait d'expliquer les choix des participants. Une approche pour mener une telle étude contrÃťlée par l'utilisateur serait d'utiliser une infrastructure de crowdsourcing comme Mechanical Turk d'Amazon.

B.3.2.3 Explications Liées

Les explications liées nécessitent des triplets réifiables. Nous utilisons les graphes nommés pour réifier triplets de données et regrouper explication. Actuellement, les meilleures pratiques en matière de publication de graphes nommés de données liées n'ont pas été finalisées. Cependant, suite à l'adoption de graphes nommés dans RDF 1.1, il est possible qu'il y ait un jour un consensus de la communauté sur les meilleures pratiques pour la publication des graphes nommés dans les données liées.

Le coût de calcul et d'accès aux explications liées dans notre approche pourrait devenir très grand. Par conséquent, les techniques de stockage et d'interrogation efficaces et notamment sous forme de graphes nommés seraient utiles. En ce sens, il existe une vaste littérature sur le stockage dynamique, l'indexation et l'interrogation pour RDF. Ces approches existantes peuvent être utilisées pour stocker et servir la grande quantité d'explication concernant les métadonnées. Nous aimerions explorer comment nous pouvons représenter et présenter efficacement les explications et leurs résumés en utilisant différents types d'interfaces utilisateur et d'interactions utilisateur. Nous aimerions explorer comment nous pouvons utiliser efficacement les classements de résumé tout en présentant toutes les informations, par exemple en chisissant l'expansion ou non d'une branche d'arbre de preuve qui contient des déclarations.

Bibliography

[Agrawal 2006] Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Chris Hayworth, Shubha Nabar, Tomoe Sugihara and Jennifer Widom. *Trio: A System for Data, Uncertainty, and Lineage.* In Proceedings of the 32Nd International Conference on Very Large Data Bases, VLDB '06, pages 1151–1154. VLDB Endowment, 2006. (Cited on page 31.)

[Aha 1991] David W. Aha, Dennis Kibler and Marc K. Albert. *Instance-based learning algorithms.* Machine Learning, vol. 6, no. 1, pages 37–66, 1991. (Cited on page 54.)

[Akdere 2012] M. Akdere, U. Çetintemel, M. Riondato, E. Upfal and S.B. Zdonik. *Learning-based Query Performance Modeling and Prediction.* In Data Engineering (ICDE), 2012 IEEE 28th International Conference on, pages 390–401, 2012. (Cited on pages 23, 35, 45, 46 and 56.)

[Alani 2006] Harith Alani, Stephen Harris and Ben O'Neil. *Winnowing Ontologies Based on Application Use.* In York Sure and John Domingue, editeurs, The Semantic Web: Research and Applications, volume 4011 of *LNCS*, pages 185–199. Springer Berlin Heidelberg, 2006. (Cited on page 126.)

[Alexander 2009] K. Alexander, R. Cyganiak, M. Hausenblas and J. Zhao. *Describing Linked Datasets - On the Design and Usage of voiD, the "Vocabulary of Interlinked Datasets".* In Linked Data on the Web Workshop (LDOW09), 18th International World Wide Web Conference (WWW09), Madrid, Spain, 2009. (Cited on page 99.)

[Alexe 2006] Bogdan Alexe, Laura Chiticariu and Wang-Chiew Tan. *SPIDER: A Schema mapPIng DEbuggeR.* In Proceedings of the 32Nd International Con-

ference on Very Large Data Bases, VLDB '06, pages 1179–1182. VLDB Endowment, 2006. (Cited on page 31.)

[Altman 1992] NS Altman. *An introduction to kernel and nearest-neighbor nonparametric regression.* The American Statistician, vol. 46, no. 3, pages 175–185, 1992. (Cited on page 54.)

[Angele 2003] J. Angele, E. Moench, H. Oppermann, S. Staab and D. Wenke. *Ontology-Based Query and Answering in Chemistry: OntoNova Project Halo.* In D. Fensel, K. Sycara and J. Mylopoulos, editeurs, The Semantic Web - ISWC 2003, volume 2870 of *LNCS*, pages 913–928. Springer Berlin / Heidelberg, 2003. (Cited on pages 26, 37, 98, 124 and 126.)

[Antoniou 2007] G. Antoniou, A. Bikakis, N. Dimaresis, M. Genetzakis, G. Georgalis, G. Governatori, E. Karouzaki, N. Kazepis, D. Kosmadakis, M. Kritsotakis, G. Lilis, A. Papadogiannakis, P. Pediaditis, C. Terzakis, R. Theodosaki and D. Zeginis. *Proof Explanation for the Semantic Web Using Defeasible Logic.* In Z. Zhang and J. Siekmann, editeurs, Knowledge Science, Engineering and Management, volume 4798 of *Lecture Notes in Computer Science*, pages 186–197. Springer Berlin / Heidelberg, 2007. (Cited on pages 27 and 37.)

[Araújo 2007] Rudi Araújo and Helena Sofia Pinto. *Towards Semantics-based Ontology Similarity.* In Proceedings of the 2nd International Workshop on Ontology Matching (OM-2007) at ISWC-2007/ASWC-2007, 2007. (Cited on page 135.)

[Arias 2011] Mario Arias, Javier D. Fernández, Miguel A. Martínez-Prieto and Pablo de la Fuente. *An Empirical Study of Real-World SPARQL Queries.* 1st International Workshop on Usage Analysis and the Web of Data (USEWOD2011), in Conjunction with WWW 2011, 2011. (Cited on page 142.)

[Bassiliades 2007] N. Bassiliades, G. Antoniou and G. Governatori. *Proof explanation in the DR-DEVICE system.* In Proc. of 1st Int'l Conference on Web Reasoning and Rule Systems, pages 249–258. Springer, 2007. (Cited on pages 27 and 37.)

[Becker 2009] Christian Becker and Christian Bizer. *Exploring the Geospatial Semantic Web with DBpedia Mobile.* Web Semant., vol. 7, no. 4, pages 278–286, December 2009. (Cited on page 21.)

[Berners-Lee 1990] Tim Berners-Lee and R. Cailliau. *WorldWideWeb: Proposal for a HyperText Project*, November 1990. (Cited on page 10.)

[Berners-Lee 2001] Tim Berners-Lee, James Hendler and Ora Lassila. *The Semantic Web.* Scientific American, vol. 284, no. 5, pages 34–43, 2001. (Cited on page 11.)

[Berners-Lee 2006a] T. Berners-Lee. *Linked Data.* W3C Design Issues `http://www.w3.org/DesignIssues/LinkedData.html`, 2006. (Cited on pages 12 and 14.)

[Berners-Lee 2006b] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer and David Sheets. In Proceedings of the 3rd International Semantic Web User Interaction Workshop, 2006. (Cited on page 20.)

[Bhagwat 2005] Deepavali Bhagwat, Laura Chiticariu, Wang-Chiew Tan and Gaurav Vijayvargiya. *An annotation management system for relational databases.* The VLDB Journal, vol. 14, no. 4, pages 373–396, 2005. (Cited on page 31.)

[Bizer 2007] C. Bizer. *Quality-Driven Information Filtering in the Context of Web-Based Information Systems.* PhD thesis, Freie Universität Berlin, Universitätsbibliothek, 2007. (Cited on pages 25, 26, 37 and 98.)

[Bizer 2014] Chris Bizer and Richard Cyganiak. *RDF 1.1 TriG*. W3C recommendation, 2014. (Cited on page 110.)

[Bonatti 2011] P.A. Bonatti, A. Hogan, A. Polleres and L. Sauro. *Robust and Scalable Linked Data Reasoning Incorporating Provenance and Trust Annotations*. Web Semantics: Science, Services and Agents on the World Wide Web, vol. 9, no. 2, pages 165–201, 2011. (Cited on page 1.)

[Buneman 2001] Peter Buneman, Sanjeev Khanna and Wang Chiew Tan. *Why and Where: A Characterization of Data Provenance*. In Proceedings of the 8th International Conference on Database Theory, ICDT '01, pages 316–330, London, UK, UK, 2001. Springer-Verlag. (Cited on pages 29, 30 and 66.)

[Buneman 2010] Peter Buneman and Egor Kostylev. *Annotation algebras for RDFS*. In The Second International Workshop on the role of Semantic Web in Provenance Management (SWPM-10), CEUR Workshop Proceedings, 2010. (Cited on pages 32, 38 and 98.)

[Bunke 1994] H. Bunke and B.T. Messmer. *Similarity measures for structured representations*. In Stefan Wess, Klaus-Dieter Althoff and MichaelM. Richter, editeurs, Topics in Case-Based Reasoning, volume 837 of *Lecture Notes in Computer Science*, pages 106–118. Springer Berlin Heidelberg, 1994. (Cited on page 49.)

[Carroll 2005] Jeremy J. Carroll, Christian Bizer, Pat Hayes and Patrick Stickler. *Named graphs, provenance and trust*. In Proceedings of the 14th international conference on World Wide Web, WWW '05, pages 613–622, New York, NY, USA, 2005. ACM. (Cited on pages 100 and 110.)

[Chang 2011] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A library for support vector machines*. ACM Transactions on Intelligent Systems and Technology,

vol. 2, pages 27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. (Cited on page 55.)

[Cheney 2009] James Cheney, Laura Chiticariu and Wang-Chiew Tan. *Provenance in Databases: Why, How, and Where*. Found. Trends databases, vol. 1, no. 4, pages 379–474, April 2009. (Cited on pages 24, 28, 29, 65 and 74.)

[Cheng 2009] Gong Cheng and Yuzhong Qu. *Searching linked objects with falcons: Approach, implementation and evaluation*. International Journal on Semantic Web and Information Systems (IJSWIS), vol. 5, no. 3, pages 49–70, 2009. (Cited on page 20.)

[Chiticariu 2006] Laura Chiticariu and Wang-Chiew Tan. *Debugging Schema Mappings with Routes*. In Proceedings of the 32Nd International Conference on Very Large Data Bases, VLDB '06, pages 79–90. VLDB Endowment, 2006. (Cited on page 31.)

[Clarke 2009] Chris Clarke. *A Resource List Management Tool for Undergraduate Students Based on Linked Open Data Principles*. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou and Elena Simperl, editeurs, The Semantic Web: Research and Applications, volume 5554 of *Lecture Notes in Computer Science*, pages 697–707. Springer Berlin Heidelberg, 2009. (Cited on page 21.)

[Corby 2006] O. Corby, R. Dieng-Kuntz, F. Gandon and Catherine Faron-Zucker. *Searching the Semantic Web: approximate query processing based on ontologies*. Intelligent Systems, IEEE, vol. 21, no. 1, pages 20–27, 2006. (Cited on page 128.)

[Corby 2012] O. Corby, A. Gaignard, C.F. Zucker and J. Montagnat. *KGRAM Versatile Inference and Query Engine for the Web of Linked Data*. In Web Intel-

ligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on, volume 1, pages 121–128, Dec 2012. (Cited on pages 33, 39, 65 and 66.)

[Cui 2000a] Y. Cui and J. Widom. *Lineage tracing in a data warehousing system.* In Data Engineering, 2000. Proceedings. 16th International Conference on, pages 683–684, 2000. (Cited on page 31.)

[Cui 2000b] Yingwei Cui, Jennifer Widom and Janet L. Wiener. *Tracing the Lineage of View Data in a Warehousing Environment.* ACM Trans. Database Syst., vol. 25, no. 2, pages 179–227, June 2000. (Cited on pages 29 and 31.)

[Damásio 2012] Carlos Viegas Damásio, Anastasia Analyti and Grigoris Antoniou. *Provenance for SPARQL Queries.* In Proceedings of the 11th International Conference on The Semantic Web - Volume Part I, ISWC'12, pages 625–640, Berlin, Heidelberg, 2012. Springer-Verlag. (Cited on pages 32, 39, 65 and 66.)

[Dividino 2009] Renata Dividino, Sergej Sizov, Steffen Staab and Bernhard Schueler. *Querying for provenance, trust, uncertainty and other meta knowledge in RDF.* Web Semantics: Science, Services and Agents on the World Wide Web, vol. 7, no. 3, pages 204–219, 2009. (Cited on pages 33, 39, 65 and 66.)

[Doyle 2003] D. Doyle, A. Tsymbal and P. Cunningham. *A Review of Explanation and Explanation in Case-Based Reasoning.* Technical Report TCD-CS-2003-41, Trinity College Dublin, 2003. (Cited on page 24.)

[Duggan 2011] Jennie Duggan, Ugur Cetintemel, Olga Papaemmanouil and Eli Upfal. *Performance Prediction for Concurrent Database Workloads.* In Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD '11, pages 337–348, New York, NY, USA, 2011. ACM. (Cited on pages 23 and 35.)

[Eduard 2005] Hovy Eduard. *Text Summarization.* In Ruslan Mitkov, editeur, The Oxford Handbook of Computational Linguistics. Oxford University Press, 2005. (Cited on pages 131, 132 and 136.)

[Ferrand 2006] Gérard Ferrand, Willy Lesaint and Alexandre Tessier. *Explanations and Proof Trees.* Computing and Informatics, vol. 25, pages 105–125, 2006. (Cited on page 115.)

[Fielding 1999] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach and Tim Berners-Lee. *Hypertext transfer protocol–HTTP/1.1*, 1999. (Cited on page 19.)

[Flouris 2009] Giorgos Flouris, Irini Fundulaki, Panagiotis Pediaditis, Yannis Theoharis and Vassilis Christophides. *Coloring RDF Triples to Capture Provenance.* In Proceedings of the 8th International Semantic Web Conference, ISWC '09, pages 196–212, Berlin, Heidelberg, 2009. Springer-Verlag. (Cited on pages 32, 38 and 98.)

[Forcher 2010] B. Forcher, M. Sintek, T. Roth-Berghofer and A. Dengel. *Explanation-Aware System Design of the Semantic Search Engine KOIOS.* In Proc. of the the 5th Int'l. Workshop on Explanation-aware Computing, 2010. (Cited on pages 25, 26, 27, 37 and 98.)

[Frasincar 2004] Flavius Frasincar, Geert-Jan Houben, Richard Vdovjak and Peter Barna. *RAL: An Algebra for Querying RDF.* World Wide Web, vol. 7, no. 1, pages 83–109, 2004. (Cited on page 47.)

[Friedman 1977] Jerome H. Friedman, Jon Louis Bentley and Raphael Ari Finkel. *An Algorithm for Finding Best Matches in Logarithmic Expected Time.* ACM Trans. Math. Softw., vol. 3, no. 3, pages 209–226, September 1977. (Cited on page 55.)

[Gaignard 2013] Alban Gaignard. *Distributed knowledge sharing and production through collaborative e-Science platforms.* PhD thesis, UniversitÃl' Nice Sophia Antipolis, 2013. (Cited on page 76.)

[Ganapathi 2009] Archana Ganapathi, Harumi Kuno, Umeshwar Dayal, Janet L. Wiener, Armando Fox, Michael Jordan and David Patterson. *Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning.* In Proceedings of the 2009 IEEE International Conference on Data Engineering, ICDE '09, pages 592–603, Washington, DC, USA, 2009. IEEE Computer Society. (Cited on pages 23, 35, 45 and 46.)

[Glass 2011] A. Glass. *Explanation of Adaptive Systems.* PhD thesis, Stanford University, 2011. (Cited on page 24.)

[Green 2007a] Todd J. Green, Grigoris Karvounarakis and Val Tannen. *Provenance Semirings.* In Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '07, pages 31–40, New York, NY, USA, 2007. ACM. (Cited on pages 29 and 30.)

[Green 2007b] Todd J Green, Grigoris Karvounarakis, Nicholas E Taylor, Olivier Biton, Zachary G Ives and Val Tannen. *ORCHESTRA: Facilitating collaborative data sharing.* In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pages 1131–1133. ACM, 2007. (Cited on page 31.)

[Green 2009] Todd J. Green. *Containment of Conjunctive Queries on Annotated Relations.* In Proceedings of the 12th International Conference on Database Theory, ICDT '09, pages 296–309, New York, NY, USA, 2009. ACM. (Cited on page 31.)

[Gupta 2008] Chetan Gupta, Abhay Mehta and Umeshwar Dayal. *PQR: Predicting Query Execution Times for Autonomous Workload Management.* In Proceed-

ings of the 2008 International Conference on Autonomic Computing, ICAC '08, pages 13–22, Washington, DC, USA, 2008. IEEE Computer Society. (Cited on pages 23, 35, 45 and 46.)

[Hall 2009] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann and Ian H Witten. *The WEKA data mining software: an update.* SIGKDD Explorations, vol. 11, no. 1, 2009. (Cited on page 54.)

[Harth 2008] Andreas Harth, Aidan Hogan, Jürgen Umbrich and Stefan Decker. *SWSE: Objects before documents!* Billion Triple Semantic Web Challenge, 7th International Semantic Web Conference 2008 (ISWC 2008), 2008. (Cited on page 21.)

[Hartig 2007] Olaf Hartig and Ralf Heese. *The SPARQL Query Graph Model for Query Optimization.* In Proceedings of the 4th European Conference on The Semantic Web: Research and Applications, ESWC '07, pages 564–578, Berlin, Heidelberg, 2007. Springer-Verlag. (Cited on pages 44 and 47.)

[Hasan 2012a] R. Hasan and F. Gandon. *Linking justifications in the collaborative semantic web applications.* In Proc. of the 21st Int'l Conference Companion on World Wide Web, WWW '12 Companion, pages 1083–1090. ACM, 2012. (Cited on page 97.)

[Hasan 2012b] Rakebul Hasan and Fabien Gandon. *A Brief Review of Explanation in the Semantic Web.* Workshop on Explanation-aware Computing (ExaCt 2012), European Conference on Artificial Intelligence (ECAI 2012), 2012. (Cited on page 10.)

[Hasan 2014a] Rakebul Hasan. *Generating and Summarizing Explanations for Linked Data.* In Valentina Presutti, Claudia d'Amato, Fabien Gandon, Mathieu d'Aquin, Steffen Staab and Anna Tordai, editeurs, The Semantic Web: Trends and Challenges, volume 8465 of *Lecture Notes in Computer Science*,

pages 473–487. Springer International Publishing, 2014. (Cited on pages 97 and 125.)

[Hasan 2014b] Rakebul Hasan. *Predicting SPARQL Query Performance and Explaining Linked Data*. In Valentina Presutti, Claudia d'Amato, Fabien Gandon, Mathieu d'Aquin, Steffen Staab and Anna Tordai, editeurs, The Semantic Web: Trends and Challenges, volume 8465 of *Lecture Notes in Computer Science*, pages 795–805. Springer International Publishing, 2014. (Cited on pages 45, 97 and 125.)

[Hasan 2014c] Rakebul Hasan and Fabien Gandon. *A Machine Learning Approach to SPARQL Query Performance Prediction*. In IEEE/WIC/ACM International Conference on Web Intelligence 2014 (WI 2014), 2014. (Cited on page 45.)

[Hasan 2014d] Rakebul Hasan and Fabien Gandon. *Predicting SPARQL Query Performance*. In 11th Extended Semantic Web Conference, ESWC2014. 2014. Poster. (Cited on page 45.)

[Hasan 2014e] Rakebul Hasan, Kemele M. Endris and Fabien Gandon. *SPARQL Query Result Explanation for Linked Data*. Semantic Web Collaborative Spaces Workshop 2014 (SWCS 2014), 13th International Semantic Web Conference 2014 (ISWC 2014), 2014. (Cited on pages 65 and 82.)

[Haynes 2001] S.R. Haynes. *Explanation in Information Systems: A Design Rationale Approach*. PhD thesis, The London School of Economics, 2001. (Cited on page 24.)

[Heath 2011] T. Heath and C. Bizer. Linked data: Evolving the web into a global data space. Morgan & Claypool, 1st édition, 2011. (Cited on pages 12, 15, 19 and 99.)

[Herschel 2010] Melanie Herschel and Mauricio A. Hernández. *Explaining Missing Answers to SPJUA Queries.* Proc. VLDB Endow., vol. 3, no. 1-2, pages 185–196, September 2010. (Cited on page 28.)

[Horridge 2008] M. Horridge, B. Parsia and U. Sattler. *Laconic and Precise Justifications in OWL.* In Proc. of the 7th Int'l Conference on the Semantic Web, ISWC '08, pages 323–338. Springer-Verlag, 2008. (Cited on pages 27, 65 and 98.)

[Horridge 2009] Matthew Horridge, Bijan Parsia and Ulrike Sattler. *Explaining Inconsistencies in OWL Ontologies.* In Lluís Godo and Andrea Pugliese, editeurs, Scalable Uncertainty Management, volume 5785 of *Lecture Notes in Computer Science*, pages 124–137. Springer Berlin Heidelberg, 2009. (Cited on pages 28, 38 and 98.)

[Horridge 2011] Matthew Horridge. *Justification Based Explanation in Ontologies.* PhD thesis, University of Manchester, 2011. (Cited on pages 27, 38 and 98.)

[Hose 2011] Katja Hose, Ralf Schenkel, Martin Theobald and Gerhard Weikum. *Database Foundations for Scalable RDF Processing.* In Axel Polleres, Claudia dâĂŹAmato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski and Peter Patel-Schneider, editeurs, Reasoning Web. Semantic Technologies for the Web of Data, volume 6848 of *Lecture Notes in Computer Science*, pages 202–249. Springer Berlin Heidelberg, 2011. (Cited on page 143.)

[Huang 2008] Jiansheng Huang, Ting Chen, AnHai Doan and Jeffrey F. Naughton. *On the Provenance of Non-answers to Queries over Extracted Data.* Proc. VLDB Endow., vol. 1, no. 1, pages 736–747, August 2008. (Cited on page 31.)

[Huang 2011] Jiewen Huang, Daniel J Abadi and Kun Ren. *Scalable SPARQL querying of large RDF graphs.* Proceedings of the VLDB Endowment, vol. 4, no. 11, pages 1123–1134, 2011. (Cited on page 44.)

[Järvelin 2002] Kalervo Järvelin and Jaana Kekäläinen. *Cumulated gain-based evaluation of IR techniques.* ACM Trans. Inf. Syst., vol. 20, no. 4, pages 422–446, October 2002. (Cited on page 135.)

[Kagal 2011] L. Kagal, I. Jacobi and A. Khandelwal. *Gasping for AIR – why we need linked rules and justifications on the semantic web.* Technical Report MIT-CSAIL-TR-2011-023, MIT, 2011. (Cited on pages 25, 27, 37 and 99.)

[Kalyanpur 2005] Aditya Kalyanpur, Bijan Parsia, Evren Sirin and James Hendler. *Debugging Unsatisfiable Classes in OWL Ontologies.* Web Semant., vol. 3, no. 4, pages 268–293, December 2005. (Cited on pages 28, 38 and 98.)

[Kalyanpur 2007] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge and Evren Sirin. *Finding All Justifications of OWL DL Entailments.* In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber and Philippe Cudré-Mauroux, editeurs, The Semantic Web, volume 4825 of *Lecture Notes in Computer Science*, pages 267–280. Springer Berlin Heidelberg, 2007. (Cited on pages 28, 38 and 98.)

[Kaufman 1987] Leonard Kaufman and Peter Rousseeuw. *Clustering by means of medoids.* In Y. Dodge, editeur, Statistical Data Analysis based on the L1 Norm, pages 405–416. 1987. (Cited on page 52.)

[Kotowski 2010] J. Kotowski and F. Bry. *A Perfect Match for Reasoning, Explanation and Reason Maintenance: OWL 2 RL and Semantic Wikis.* In Proc. of 5th Semantic Wiki Workshop, 2010. (Cited on pages 27, 37 and 98.)

[Lam 2008] Joey Sik Chun Lam, Derek Sleeman, Jeff Z. Pan and Wamberto Vasconcelos. *A Fine-Grained Approach to Resolving Unsatisfiable Ontologies*. In Stefano Spaccapietra, editeur, Journal on Data Semantics X, volume 4900 of *Lecture Notes in Computer Science*, pages 62–95. Springer Berlin Heidelberg, 2008. (Cited on pages 28, 38 and 98.)

[Li 2010] Ning Li and Enrico Motta. *Evaluations of User-Driven Ontology Summarization*. In Philipp Cimiano and H.Sofia Pinto, editeurs, Knowledge Engineering and Management by the Masses, volume 6317 of *LNCS*, pages 544–553. Springer Berlin Heidelberg, 2010. (Cited on page 132.)

[Lim 2009] Brian Y. Lim, Anind K. Dey and Daniel Avrahami. *Why and Why Not Explanations Improve the Intelligibility of Context-aware Intelligent Systems*. In Proc. of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09, pages 2119–2128, New York, NY, USA, 2009. ACM. (Cited on pages 83 and 84.)

[Mateen 2014] Abdul Mateen, Basit Raza, Muhammad Sher, M.M. Awais and Norwatti Mustapha. *Workload management: a technology perspective with respect to self-* characteristics*. Artificial Intelligence Review, vol. 41, no. 4, pages 463–489, 2014. (Cited on page 2.)

[McGuinness 2003] Deborah L. McGuinness and Paulo Pinheiro Silva. *Infrastructure for Web Explanations*. In Dieter Fensel, Katia Sycara and John Mylopoulos, editeurs, The Semantic Web - ISWC 2003, volume 2870 of *LNCS*, pages 113–129. Springer Berlin Heidelberg, 2003. (Cited on pages 26, 82, 96, 97, 125 and 140.)

[McGuinness 2004] D.L. McGuinness and P Pinheiro da Silva. *Explaining answers from the Semantic Web: the Inference Web approach*. Web Semantics: Science, Services and Agents on the World Wide Web, vol. 1, no. 4, pages 397 – 413, 2004. (Cited on pages 25, 26, 82, 97, 124, 125 and 126.)

[McGuinness 2006] D.L. McGuinness, Li Ding, A. Glass, C. Chang, H. Zeng and V. Furtado. *Explanation Interfaces for the Semantic Web: Issues and Models.* In Proc. of the 3rd Int'l Semantic Web User Interaction Workshop, 2006. (Cited on page 25.)

[McGuinness 2008] D.L. McGuinness, V. Furtado, P. Pinheiro da Silva, L. Ding, A. Glass and C. Chang. *Explaining Semantic Web Applications.* In Semantic Web Engineering in the Knowledge Society. 2008. (Cited on pages 25, 26, 37, 65, 97 and 125.)

[McSherry 2008] Frank McSherry and Marc Najork. *Computing Information Retrieval Performance Measures Efficiently in the Presence of Tied Scores.* In Craig Macdonald, Iadh Ounis, Vassilis Plachouras, Ian Ruthven and RyenW. White, editeurs, Advances in Information Retrieval, volume 4956 of *LNCS*, pages 414–421. Springer Berlin Heidelberg, 2008. (Cited on page 135.)

[Meliou 2009] Alexandra Meliou, Wolfgang Gatterbauer, Katherine F Moore and Dan Suciu. *Why So? or Why No? Functional Causality for Explaining Query Answers.* Technical Report 09-12-01, University of Washington, 2009. (Cited on page 31.)

[Meyer 2006] Thomas Meyer, Kevin Lee, Richard Booth and Jeff Z. Pan. *Finding Maximally Satisfiable Terminologies for the Description Logic ALC.* In Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1, AAAI'06, pages 269–274. AAAI Press, 2006. (Cited on pages 28, 38 and 98.)

[Moodley 2010] Kodylan Moodley. *Debugging and repair of Description Logic ontologies*, 2010. (Cited on pages 28, 38 and 98.)

[Moore 1988] J.D. Moore and W.R. Swartout. *Explanation in Expert Systems: A Survey.* Research Report ISI/RR-88-228, University of Southern California, 1988. (Cited on page 24.)

[Moreau 2013] L. Moreau and P. Missier. *PROV-DM: The PROV Data Model*. World Wide Web Consortium, W3C Recommendation, 2013. (Cited on page 31.)

[Morsey 2011] Mohamed Morsey, Jens Lehmann, Sören Auer and Axel-Cyrille Ngonga Ngomo. *DBpedia SPARQL Benchmark - Performance Assessment with Real Queries on Real Data*. In Lora Aroyo*et al.*, editeurs, The Semantic Web âĂŞ ISWC 2011, volume 7031 of *LNCS*, pages 454–469. Springer Berlin Heidelberg, 2011. (Cited on pages 53 and 73.)

[Nandi 2007] Arnab Nandi and H. V. Jagadish. *Assisted querying using instant-response interfaces*. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data, SIGMOD '07, pages 1156–1158, New York, NY, USA, 2007. ACM. (Cited on pages 23 and 34.)

[Owens 2008] Alisdair Owens, Andy Seaborne, Nick Gibbins and mc schraefel. Clustered TDB: A clustered triple store for Jena. November 2008. (Cited on pages 53 and 73.)

[OWL 2014] *OWL Web Ontology Language Overview*. W3C recommendation, 2014. (Cited on page 18.)

[Peroni 2008] Silvio Peroni, Enrico Motta and Mathieu d'Aquin. *Identifying Key Concepts in an Ontology, through the Integration of Cognitive Principles with Statistical and Topological Measures*. In John Domingue and Chutiporn Anutariya, editeurs, The Semantic Web, volume 5367 of *LNCS*, pages 242–256. Springer Berlin Heidelberg, 2008. (Cited on page 125.)

[Pinheiro da Silva 2006] P. Pinheiro da Silva, D.L. McGuinness and R. Fikes. *A Proof Markup Language for Semantic Web Services*. Information Systems, vol. 31, no. 4-5, pages 381–395, 2006. (Cited on pages 25, 65 and 99.)

[Pinheiro da Silva 2008] P. Pinheiro da Silva, D.L. McGuinness, N. Del Rio and L. Ding. *Inference Web in Action: Lightweight Use of the Proof Markup Language*. In Proc. of the 7th Int'l Semantic Web Conference, ISWC '08, pages 847–860, 2008. (Cited on pages 25, 83 and 99.)

[RDF 2014a] *RDF 1.1 Concepts and Abstract Syntax*. W3C recommendation, 2014. (Cited on pages 12 and 100.)

[RDF 2014b] *RDF 1.1 Semantics*. W3C recommendation, 2014. (Cited on page 99.)

[RDF 2014c] *RDF Schema 1.1*. W3C recommendation, 2014. (Cited on page 18.)

[Riesen 2009] Kaspar Riesen and Horst Bunke. *Approximate graph edit distance computation by means of bipartite graph matching*. Image Vision Comput., vol. 27, no. 7, pages 950–959, June 2009. (Cited on pages 49, 50 and 51.)

[Riesen 2013] Kaspar Riesen, Sandro Emmenegger and Horst Bunke. *A Novel Software Toolkit for Graph Edit Distance Computation*. In WalterG. Kropatsch, NicoleM. Artner, Yll Haxhimusa and Xiaoyi Jiang, editeurs, Graph-Based Representations in Pattern Recognition, volume 7877 of *LNCS*, pages 142–151. Springer Berlin Heidelberg, 2013. (Cited on page 51.)

[Roth-Berghofer 2004] T. Roth-Berghofer. *Explanations and Case-Based Reasoning: Foundational Issues*. In P. Funk and P.A.G. Calero, editeurs, Advances in Case-Based Reasoning, pages 389–403. Springer-Verlag, 2004. (Cited on page 24.)

[Schlobach 2003] Stefan Schlobach and Ronald Cornet. *Non-standard Reasoning Services for the Debugging of Description Logic Terminologies*. In Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI'03, pages 355–360, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc. (Cited on pages 28, 38 and 98.)

[Schwarte 2011] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel and Michael Schmidt. *Fedx: Optimization techniques for federated query processing on linked data.* In Proc. of the 10th International Semantic Web Conference, ISWC 2011. Springer, 2011. (Cited on pages 1, 44, 76 and 141.)

[Shevade 2000] Shirish Krishnaj Shevade, S Sathiya Keerthi, Chiranjib Bhattacharyya and Karaturi Radha Krishna Murthy. *Improvements to the SMO algorithm for SVM regression.* Neural Networks, IEEE Transactions on, vol. 11, no. 5, pages 1188–1193, 2000. (Cited on page 55.)

[Shinavier 2010] Joshua Shinavier. *Position paper: Named Graphs in Linked Data.* 2010. (Cited on page 143.)

[Simperl 2013] E. Simperl, B. Norton, M. Acosta, M. Maleshkova, J. Domingue, A. Mikroyannidis, P. Mulholland and R. Power. Using linked data effectively. The Open University, Milton Keynes, December/Winter 2013. (Cited on page 11.)

[SKO 2009] *SKOS Simple Knowledge Organization System Reference.* W3C recommendation, 2009. (Cited on page 17.)

[SPA 2013a] *SPARQL 1.1 Protocol.* W3C recommendation, 2013. (Cited on page 19.)

[SPA 2013b] *SPARQL 1.1 Query Language.* W3C recommendation, 2013. (Cited on pages 12 and 13.)

[Steinberger 2009] Josef Steinberger and Karel Jezek. *Evaluation Measures for Text Summarization.* Computing and Informatics, vol. 28, no. 2, pages 251–275, 2009. (Cited on pages 132 and 137.)

[Stocker 2008] Markus Stocker, Andy Seaborne, Abraham Bernstein, Christoph Kiefer and Dave Reynolds. *SPARQL Basic Graph Pattern Optimization*

Using Selectivity Estimation. In Proceedings of the 17th International Conference on World Wide Web, WWW '08, pages 595–604, New York, NY, USA, 2008. ACM. (Cited on pages 40, 46, 47 and 142.)

[Stojanovic 2004] Nenad Stojanovic and Ljiljana Stojanovic. *A Logic-Based Approach for Query Refinement in Ontology-Based Information Retrieval S.* In Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, ICTAI '04, pages 450–457, Washington, DC, USA, 2004. IEEE Computer Society. (Cited on pages 23 and 34.)

[Stumpf 2007] S. Stumpf, V. Rajaram, L. Li, M. Burnett, T. Dietterich, E. Sullivan, R. Drummond and J. Herlocker. *Toward harnessing user feedback for machine learning.* In Proc. of the 12th international conference on Intelligent user interfaces, IUI '07, pages 82–91. ACM, 2007. (Cited on page 24.)

[Swartout 1991] W. Swartout, C. Paris and J. Moore. *Explanations in knowledge systems: design for explainable expert systems.* IEEE Expert, vol. 6, no. 3, pages 58 –64, 1991. (Cited on page 24.)

[Theoharis 2011] Yannis Theoharis, Irini Fundulaki, Grigoris Karvounarakis and Vassilis Christophides. *On Provenance of Queries on Semantic Web Data.* IEEE Internet Computing, vol. 15, no. 1, pages 31–39, January 2011. (Cited on pages 2, 28, 32, 39, 64, 65 and 66.)

[Tintarev 2007] N. Tintarev and J. Masthoff. *A Survey of Explanations in Recommender Systems.* ICDE'07 Workshop on Recommender Systems and Intelligent User Interfaces, 2007. (Cited on page 24.)

[Tintarev 2012] Nava Tintarev and Judith Masthoff. *Evaluating the Effectiveness of Explanations for Recommender Systems.* User Modeling and User-Adapted Interaction, vol. 22, no. 4-5, pages 399–439, October 2012. (Cited on page 83.)

[Tran 2010] Quoc Trung Tran and Chee-Yong Chan. *How to ConQueR Why-not Questions*. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10, pages 15–26, New York, NY, USA, 2010. ACM. (Cited on page 31.)

[Tsialiamanis 2012] Petros Tsialiamanis, Lefteris Sidirourgos, Irini Fundulaki, Vassilis Christophides and Peter Boncz. *Heuristics-based Query Optimisation for SPARQL*. In Proceedings of the 15th International Conference on Extending Database Technology, EDBT '12, pages 324–335, New York, NY, USA, 2012. ACM. (Cited on pages 35, 40, 44, 46 and 47.)

[Tummarello 2007] Giovanni Tummarello, Renaud Delbru and Eyal Oren. *Sindice.Com: Weaving the Open Linked Data*. In Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference, ISWC'07/ASWC'07, pages 552–565, Berlin, Heidelberg, 2007. Springer-Verlag. (Cited on page 21.)

[Tummarello 2010] Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, Renaud Delbru and Stefan Decker. *Sig.Ma: Live Views on the Web of Data*. In Proceedings of the 19th International Conference on World Wide Web, WWW '10, pages 1301–1304, New York, NY, USA, 2010. ACM. (Cited on page 20.)

[Wang 1990] Y. Richard Wang and Stuart E. Madnick. *A Polygon Model for Heterogeneous Database Systems: The Source Tagging Perspective*. In Proceedings of the Sixteenth International Conference on Very Large Databases, pages 519–533, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc. (Cited on pages 29 and 30.)

[Wang 2005] Hai Wang, Matthew Horridge, Alan Rector, Nick Drummond and Julian Seidenberg. *Debugging OWL-DL Ontologies: A Heuristic Approach*. In

Yolanda Gil, Enrico Motta, V.Richard Benjamins and MarkA. Musen, editeurs, The Semantic Web – ISWC 2005, volume 3729 of *Lecture Notes in Computer Science*, pages 745–757. Springer Berlin Heidelberg, 2005. (Cited on pages 28, 38 and 98.)

[Wylot 2014] Marcin Wylot, Philippe Cudre-Mauroux and Paul Groth. *TripleProv: Efficient Processing of Lineage Queries in a Native RDF Store*. In Proceedings of the 23rd International Conference on World Wide Web, WWW '14, pages 455–466, 2014. (Cited on pages 2, 28, 32, 33, 39, 64, 65, 66 and 74.)

[Zenz 2009] Gideon Zenz, Xuan Zhou, Enrico Minack, Wolf Siberski and Wolfgang Nejdl. *From keywords to semantic queries-Incremental query construction on the semantic web*. Web Semant., vol. 7, no. 3, pages 166–176, September 2009. (Cited on pages 23 and 34.)

[Zhang 2007] Xiang Zhang, Gong Cheng and Yuzhong Qu. *Ontology summarization based on RDF sentence graph*. In Proceedings of the 16th international conference on World Wide Web, WWW '07, pages 707–716, New York, NY, USA, 2007. ACM. (Cited on pages 125, 130, 131 and 134.)

[Zimmermann 2012] Antoine Zimmermann, Nuno Lopes, Axel Polleres and Umberto Straccia. *A General Framework for Representing, Reasoning and Querying with Annotated Semantic Web Data*. Web Semant., vol. 11, pages 72–95, March 2012. (Cited on pages 33, 39, 65 and 66.)