ON TECHNIQUES FOR PAY-AS-YOU-GO DATA INTEGRATION OF LINKED DATA

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy in the Faculty of Engineering and Physical Sciences

2014

By
Klitos Christodoulou
School of Computer Science
# Contents

Abstract 10  
Declaration 12  
Copyright 13  
Acknowledgements 14  
Glossary and Acronyms 17  

1 Introduction 18  
1.1 Setting the Scene 18  
1.2 Motivation 20  
1.3 Dataspaces 22  
1.4 Aim, Objectives and Research Contributions 25  
1.5 Overview of Thesis Structure 27  

2 Anatomy of Dataspaces over Linked Data 29  
2.1 The Semantic Web Vision 30  
2.2 The Linked Data Paradigm 32  
2.2.1 Resource Identification with URIs 37  
2.2.2 Dereferenceable URIs 37  
2.2.3 A Uniform Data Model: RDF 38  
2.2.4 Include Links to Other URIs 42  
2.3 Vocabularies, RDF Schema and OWL 44  
2.4 SPARQL: Querying RDF 47  
2.5 Data Integration in Linked Data 50  
2.5.1 Federated Query Processing 52  
2.5.2 WoD as a Global Distributed Dataspase 53
2.5.3 Aggregation of Search Results ........................................... 56
2.6 Dataspaces over Linked Data ............................................. 57
  2.6.1 Life-cycle of a Dataspaces over LD ................................. 60
  2.6.2 Uncertainty in Dataspaces ........................................... 75
2.7 Discussion and Conclusions ................................................. 79

3 Structure Inference for Linked Data Sources ................................. 81
  3.1 An Overview of Clustering Techniques ................................. 82
  3.2 An Overview of the Contributed Approach ........................... 87
  3.3 Schema Inference using Clustering ................................... 90
    3.3.1 Identify Input RDF Graph .................................... 90
    3.3.2 Pre-processing .................................................. 93
    3.3.3 Cluster Candidate Descriptions ............................... 95
    3.3.4 Annotation of Clusters ....................................... 101
    3.3.5 Infer a Conceptual Structure ................................ 107
  3.4 Experimental Evaluation ............................................... 109
    3.4.1 Experiment 1: Reverse Engineering ............................ 114
    3.4.2 Experiment 2: Sources from the Web of Data ................. 120
  3.5 Related Work .......................................................... 122
    3.5.1 Source Discovery ............................................... 123
    3.5.2 Knowledge Discovery and Ontology Mining .................... 124
    3.5.3 Distributed Query Processing ................................. 126
  3.6 Discussion and Conclusions ............................................. 126

4 Managing uncertainty in LD matching ........................................... 129
  4.1 Overview of Probability ............................................... 132
    4.1.1 Probability Theory and Axioms ................................ 132
  4.2 Random Variables and Probability Distributions ..................... 134
  4.3 Conditional Probability Distributions ............................... 135
  4.4 Bayesian Inference .................................................... 136
  4.5 Overview of Approach .................................................. 138
  4.6 Similarity Scores to Degrees of Belief ............................... 144
    4.6.1 Empirical Study: Learning from Similarity Scores ............ 146
    4.6.2 The Derivation of Score Distributions ......................... 152
    4.6.3 Updating Degrees of Belief given Similarity Scores .......... 159
  4.7 Semantic Annotations to Degrees of Belief ........................... 165
List of Tables

2.1 Anatomy of an RDF statement. ................. 41
2.2 Summary of model-generic and model-specific constructs. ....... 62

3.1 Suggested classes from a cluster’s metadata annotations. ...... 107
3.2 Suggested properties from a cluster’s metadata annotations. ... 108
3.3 Suggested relationships from a cluster’s meta-data annotations. . 108
3.4 Linked data sources participating in evaluation. ................. 109
3.5 Quality of inferred schema, where (ET): Entity Types, (AT): Attributes, and (R): Relationships. ................. 119

4.1 Different kinds of evidence in relation to matching that can be assimilated by the Bayesian Updating Framework. ................. 141
4.2 Rules used to derive the set of equivalent pairs for classes/properties. 177
4.3 Rules used to derive the set of non-equivalent pairs for classes/properties. ............................... 177
4.4 Mean conditional probabilities assigned to each semantic evidence (equivalent) classes. .................................................. 178
4.5 Mean conditional probabilities assigned to each semantic evidence (non-equivalent) classes. ............................... 179
4.6 Participating ontologies from OAEI conference track. .......... 186
4.7 AVG scheme vs. Bayesian syntactic. ............................... 188
4.8 AVG scheme vs. Bayesian syntactic & semantic. ................ 189
4.9 Bayesian syntactic vs. Bayesian syntactic & semantic. .......... 191

B.1 Description of different types of mutations on local-names. ....... 223
B.2 Probability of occurrence of mutations for different configurations. 224

D.1 Similarity score observations organised to produce a histogram for n-gram (equivalent case). .................................................. 228
G.1 Mappings of prefixes used. 231
List of Figures

1.1 Examples of instance level data. ........................................... 21

2.1 RDF sub-graph from Magnatune with sample annotations at the conceptual level. ...................................................... 36

2.2 Integration schema used for the integration of RDF data sources from DBTune.org. .................................................... 63

2.3 Inferred conceptual descriptions of DBTune.org RDF data sources represented as entity-relationship diagrams. .................. 68

2.4 Schema matching results. ................................................... 70

2.5 Schematic correspondences inference. ................................ 70

2.6 Result tuples annotated with user feedback. ......................... 74

3.1 Abstract description of schema inference methodology. ........... 89

3.2 Interconnected RDF sub-graph extracted from the Jamendo dataset. 91

3.3 Example of an individual represented as a candidate description. 94

3.4 Example of a dendrogram with possible cutting levels at different dissimilarities. ....................................................... 101

3.5 Choice of $t$ and its effect on the maximum average silhouette width. 113

3.6 Different linkage schemes, the effect on the top-k maximum average silhouette width and the suggested number of clusters. ....... 115

3.7 Quality of resulted clustering using CDShop data source. ......... 117

3.8 Quality of resulted clustering using Conference data source. ...... 117

3.9 Quality of resulted clustering using Birt_DB data source. ......... 117

3.10 Quality of inferred schema for Jamendo. ............................... 121

3.11 Quality of inferred schema for Magnatune. ............................ 121

4.1 The assimilation of different pieces of evidence and the update of degree of belief using the Bayesian Updating Framework. ...... 138

4.2 Examples of URIs with their local-names. ............................. 147
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>Experimental methodology for the construction of an estimated density function from a sample of observations.</td>
<td>148</td>
</tr>
<tr>
<td>4.4</td>
<td>Example of matches returned by using the <em>edit-distance</em> matcher on pair of local names.</td>
<td>151</td>
</tr>
<tr>
<td>4.5</td>
<td>Classification of matches to true/false positive matches and the derivation of observations using edit-distance.</td>
<td>151</td>
</tr>
<tr>
<td>4.6</td>
<td>The impact of <em>bin width</em> over the histograms constructed with the set $E$ of similarity score observations for the <em>n-gram</em> matcher.</td>
<td>153</td>
</tr>
<tr>
<td>4.7</td>
<td>Histograms showing the distribution of similarity scores for string-based matchers.</td>
<td>154</td>
</tr>
<tr>
<td>4.8</td>
<td>Direct application of kernel density estimate over similarity scores derived by the tri-gram matcher ($n = 679$, $h = 0.038$, <em>Gaussian kernel</em>) with support $(-\infty, +\infty)$.</td>
<td>156</td>
</tr>
<tr>
<td>4.9</td>
<td>Boundary corrected kernel density estimate over similarity scores derived by the tri-gram matcher ($n = 679$, $h = 0.488$, <em>Gaussian kernel</em>) with support $[0,1]$.</td>
<td>156</td>
</tr>
<tr>
<td>4.10</td>
<td>Boundary corrected kernel density estimates derived from similarity score observations for the string-based matchers using a <em>Gaussian kernel</em> and support $[0,1]$.</td>
<td>158</td>
</tr>
<tr>
<td>4.11</td>
<td>Bayesian updating with the assimilation of similarity scores returned by the matchers.</td>
<td>161</td>
</tr>
<tr>
<td>4.12</td>
<td>Effect on the posterior probabilities for syntactic evidence assuming different priors.</td>
<td>164</td>
</tr>
<tr>
<td>4.13</td>
<td>Pieces of syntactic and semantic knowledge at the conceptual level between a pair of RDFS/OWL classes.</td>
<td>166</td>
</tr>
<tr>
<td>4.14</td>
<td>A lightweight model for random crawling the Web of Data.</td>
<td>170</td>
</tr>
<tr>
<td>4.15</td>
<td>Number of N-Quads retrieved per crawl round.</td>
<td>172</td>
</tr>
<tr>
<td>4.16</td>
<td>Vocabulary elements of RDFS/OWL and SKOS.</td>
<td>174</td>
</tr>
<tr>
<td>4.17</td>
<td>Examples of posterior probabilities computed for different semantic evidence.</td>
<td>180</td>
</tr>
<tr>
<td>4.18</td>
<td>Example of a survey question.</td>
<td>183</td>
</tr>
<tr>
<td>4.19</td>
<td>Regions of interest used for evaluation.</td>
<td>187</td>
</tr>
<tr>
<td>4.20</td>
<td>Individual absolute errors showing AVG scheme vs. Bayesian syntactic and semantic.</td>
<td>189</td>
</tr>
</tbody>
</table>
4.21 Individual absolute errors showing Bayesian syntactic vs. Bayesian syntactic and semantic. ................................. 190

E.1 Comparison of density estimates for similarity score observations using a standard Gaussian kernel with the transformation Gaussian kernel that has bounded support [0,1]. .............................. 229
Abstract

It is recognised that nowadays, users interact with large amounts of data that exist in disparate forms, and are stored under different settings. Moreover, it is true that the amount of structured and un-structured data outside a single well organised data management system is expanding rapidly. To address the recent challenges of managing large amounts of potentially distributed data, the vision of a dataspace was introduced. This data management paradigm aims at reducing the complexity behind the challenges of integrating heterogeneous data sources.

Recently, efforts by the Linked Data (LD) community gave rise to a Web of Data (WoD) that interweaves with the current Web of documents in a way that it is useful for data consumption by both humans and computational agents. On the WoD, datasets are structured under a common data model and published as Web resources following a simple set of guidelines that enables them to be linked with other pieces of data, as well as, to be annotated with useful meta data that help determine their semantics. The WoD is an evolving open ecosystem including specialist publishers as well as community efforts aiming at re-publishing isolated databases as LD on the WoD, and annotating them with meta data.

The WoD raises new opportunities and challenges. However, currently it mostly relies on manual efforts for integrating the large amounts of heterogeneous data sources on the WoD. This dissertation makes the case that several techniques from the dataspaces research area (aiming at on-demand integration of data sources in a pay-as-you-go fashion) can support the integration of heterogeneous WoD sources. In so doing, this dissertation explores the opportunities and identifies the challenges of adapting existing pay-as-you-go data integration techniques in the context of LD. More specifically, this dissertation makes the following contributions: (i) a case-study for identifying the challenges when existing pay-as-you-go data integration techniques are applied in a setting where data sources are LD; (ii) a methodology that deals with the “schema-less” nature of
LD sources by automatically inferring a conceptual structure from a given RDF graph thus enabling downstream tasks, such as the identification of matches and the derivation of mappings, which are, both, essential for the automatic bootstrapping of a dataspace; and (iii) a well-defined, principled methodology that builds on a Bayesian inference technique for reasoning under uncertainty to improve pay-as-you-go integration. Although the developed methodology is generic in being able to reason with different hypothesis, its effectiveness has only been explored on reducing the uncertain decisions made by string-based matchers during the matching stage of a dataspace system.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on presentation of Theses.
Acknowledgements

“As you set out for Ithaka hope the voyage is a long one, full of adventure, full of discovery.”

Constantinos C. Cavafy, Poet Obtaining, 1911.

Obtaining a doctoral degree is a fascinating journey through knowledge; full of new discoveries as well as surprises along the way. Although mine was not an easy endeavour, the knowledge obtained along my voyage has been absolutely worthwhile. Throughout this journey I was never left alone, and therefore I would like to express my deepest gratitude to those who helped me in one way or another to reach my destination.

I enjoyed the rare privilege of being supervised by two distinguished researchers, Dr. Alvaro A. A. Fernandes and Prof. Norman W. Paton, who taught me how to conduct research, how to ground research problems and to choose important ones to solve. Their constant support, guidance, friendship, patience and encouragement always kept me on the right path. Your help was invaluable; thank you for having me as your student.

I am also extremely grateful to another set of individuals who have helped me grow as a researcher; a big thank you to Dr. Alasdair J. G. Gray for his support and help, especially during the first days of my research, Dr. Cornelia Hedeler who I worked closely for extending the DSToolkit project and for her great support along the way and Dr. Khalid Belhajjame for his friendly advice, support and guidance throughout my research studies.

The research outcomes reported in this dissertation would not have materialised without the funding supplied by the Engineering and Physical Sciences Research Council (EPSRC) whose financial support I am pleased to acknowledge.

Furthermore, many thanks must go to the wonderful members of the department and especially to the Information Management Group (IMG) at the School
of Computer Science at the University of Manchester for the lovely coffee breaks, social events, and discussions. Special thanks also go to my friends and office mates, Alan Stokes, Fernando Osorno, Duhai Alshukaili, Julio Cesar, Ruhaila Maskat and many more whose support, distractions, and discussions kept me going through tough times.

Moreover, my thanks go to my compatriot friends for the fun nights out and discussions related to research and life in general. I am profoundly grateful, and send my special thanks to my great friend and flatmate Andreas for his great friendship and for enriching my mathematical knowledge.

My boundless love and appreciation goes to my wonderful parents, Andreas and Theodosia, for their constant love, continuous encouragement and unwavering support through this journey. Mum and Dad, thank you for supporting me during the more stressful periods and for believing in me. I am forever grateful for everything that you have done for me, and I would like to dedicate this dissertation to you. Last but not least, I would like to express my heartfelt gratitude to my brother Panayiotis for his unconditional support and for encouraging me to pursue my interests.

Manchester, August 2014
Klitos Christodoulou
Αφιερωμένο στους γονείς μου.
To my parents.
**Glossary and Acronyms**

**BUF** Bayesian Updating Framework.

**DSMS** Dataspace Management System.

**DSToolkit** A dataspace management system that supports pay-as-you-go data integration [Hedeler et al., 2012].

**HTML** HyperText Markup Language.

**HTTP** HyperText Transfer Protocol.

**KDE** Kernel Density Estimation.

**LD** Linked Data.

**p.d.f.** probability density function.

**p.m.f.** probability mass function.

**probabilistic model** A probabilistic model is a mathematical description of an uncertain phenomenon [Bertsekas and Tsitsiklis, 2008].

**RDF** Resource Description Framework.

**SW** Semantic Web.

**URI** Uniform Resource Identifier.

**URL** Uniform Resource Locator.

**WoD** Web of Data.
Chapter 1

Introduction

“You can have data without information, but you cannot have information without data.”

Daniel Keys Moran

1.1 Setting the Scene

There has been a general trend towards generating large amounts of data [Doan et al., 2009], which is made available under different settings. The World Wide Web has been widely used as a medium for publishing information for years; large enterprises and organisations run different but coexisting information systems that store large volumes of information, and even personal information systems may deal with large amounts of information. Information is often stored in data sources that are typically distributed and frequently designed from independent processes or actors, and maintained autonomously.

Recently, there has been a rapid growth in the amount of structured data published on the Web. An enabler of this transition from the classic document-oriented Web into a Web of interlinked data is the paradigm of Linked Data [Bizer et al., 2009a]. In essence, the term Linked Data (LD) refers to a set of principles that involve the publication, and interlinking of structured data on the Web. These principles have been motivated by the need to access Web data stored in different, isolated, “data silos” using various data models or formats, and different interfaces (e.g., Web APIs) [Bizer, 2009].

Initially proposed by Berners-Lee [2006], the LD principles (paraphrasing [Berners-Lee, 2006]) suggest: (i) the use of URIs as names of things; (ii) using
HTTP URIs so that things can be looked up (i.e., dereferenced); (iii) returning, upon lookup useful information in standard representations (RDF, SPARQL); and (iv) including links, by way of URIs to other related RDF documents, to allow further navigation.

The simplicity of the principles had a major impact on their adoption, encouraging a variety of data publishers or third-parties to publish their datasets in RDF on the Web, and to interlink them with each other, leading to the emergence of a global data space known as the Web of Data (WoD).

Evidence of the emergence of the WoD comes from the Linking Open Data project\(^1\), a community effort with the aim of identifying and re-publishing open licence data sources using the linked data principles as RDF graphs on the Web. The results from such community-driven efforts are reflected in the Linking Open Data (LOD) “cloud”\(^2\), which contains billions of RDF triples from diverse domains, such as geographic information (e.g., Geonames\(^3\)), academia (e.g., DBLP\(^4\)), music (e.g., DBTune\(^5\)), government-related data (e.g., data.gov.uk), context from Wikipedia (e.g., DBpedia\(^6\)), biomedical datasets (e.g., DrugBank\(^7\)), and many more. In reality, the WoD extends the LOD cloud into an ecosystem of interlinked data published on the Web\(^8\):

> “the WoD is generic and can contain any type of data; anyone can publish data to the Web of Data; data publishers are not constrained in the choice of vocabularies with which to represent the data; entities are connected by RDF links.” Bizer et al. [2009a].

Indeed, LD supports publication activities by providing low barriers to entry, whereby publishers can publish their data sources using a small set of Semantic Web technologies. Consequently, the WoD can be characterised as an evolving ecosystem of data sources (i.e., RDF graphs) that are heterogeneous and interlinked. Since its inception, numerous publishers from a variety of domains have made their datasets available for public use in a setting that is something of

---
\(^1\)http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
\(^2\)http://lod-cloud.net/
\(^3\)http://geonames.org/
\(^4\)http://www4.wiwiss.fu-berlin.de/dblp/
\(^5\)http://dbtune.org/
\(^6\)http://dbpedia.org/
\(^7\)http://www.drugbank.ca/
\(^8\)To get an idea of the scale of the WoD visit the LODStats project http://stats.lod2.eu/
CHAPTER 1. INTRODUCTION

a free-for-all. However, there are several existing challenges that relate to the consumption of the WoD that are the subject of ongoing research.

One question that arises and that lies at heart of this dissertation is: could existing data management techniques be adopted or adapted for use in this new context, thereby enabling consumers to deal with challenges such as the integration of heterogeneous and autonomous LD sources?

1.2 Motivation

The WoD has several unique characteristics that could perhaps be utilised to ease the integration of large numbers of LD sources. In particular, the LD principles guarantee uniformity in terms of a common homogeneous model for publishing data in the form of subject-predicate-object RDF triples. Real world entities are represented by global unique identifiers (i.e., URIs), where de-referenceability allows one to discover more information (e.g., about the predicates used to describe a resource), or to get access to semantic annotations of classes or properties (i.e., schema-level terminology) from LD vocabularies. In general, the fact that data are interlinked on the WoD enables a follow-your-nose style of data and meta-information discovery. Importantly, the use of Semantic Web languages, such as the RDFS [Brickley and Guha, 2004] and OWL [Mcguinness and van Harmelen, 2004], allow data items to be annotated with rich semantics which, for example, can be used to reconcile various kinds of heterogeneities between LD vocabularies (such annotations are likely to contain useful knowledge e.g., owl:equivalentClass link, in Figure 1.1). Additionally, SPARQL [Prud’hommeaux and Seaborne, 2008] is a standard query language for RDF that offers opportunities for creating view-base querying of integrated resources.

Simultaneously, the WoD can be a challenging environment for consumers seeking a coherent and integrated view of data from LD sources. Before accounting for this, this dissertation envisions the WoD as having two levels: the instance level, where URIs are used to uniquely identify real-world entities (using terminology from ontology research, this is the assertional-level known as the A-Box level), and the conceptual level, in the form of various LD vocabularies that are used to define the semantic meaning over the published data from LD sources (known as the T-Box level in ontology research). Specifically,
the WoD can be a highly heterogeneous environment where the heterogeneity stems from both these levels. To see this consider Figure 1.1. There is an incomplete agreement at the instance level in terms of a unique URI to represent every real world entity under the same URI. For example, the URIs http://dbpedia.org/page/Greece and http://linkeddata.org/page/triplify/node432424989 are both used to describe Greece. Similarly, at the conceptual level, the vocabulary URIs of classes http://dbpedia.org/ontology/Country and http://linkeddata.org/ontology/Country can both be used to represent the same real world concept. Moreover, there is inconsistency/contradiction in the values of predicates of those URIs used to describe properties of the same real world entities. As an example, consider the inconsistency related to the population of Greece from dbpedia and linkeddata provided by the values of dbpprop:populationCensus and lgdo:population respectively.

Moreover, consumers of LD are likely to deal with heterogeneity in terminology. As an example consider, the predicate dbpprop:commonName or lgdo:internationalName both defined to capture the name of a country. As in this example, LD publishers are likely to use different terminology to semantically annotate their data. This stems from two facts: (i) there is no restriction on the terms used or which vocabulary to use; publishers have the flexibility to “cherry-pick” terms from various vocabularies [Heath and Bizer, 2011]; and (ii) the fact that publishers are free to create vocabularies with different granularity or expressiveness in order to satisfy their publication needs. Frequently, the decision as to which vocabulary or set of terms a publisher should use may be influenced by the community of which the publisher is a member.
Furthermore, the LD sources do not adhere to a specific conceptualisation description. According to Heath and Bizer [2011]:

“the term schema is understood in the Linked Data context as the mixture of distinct terms from different RDF vocabularies that are used by a data source to publish data on the Web. This mixture may include terms from widely used vocabularies as well as proprietary terms.”

Although publication practices for LD advocate the reuse of terms from well-known vocabularies (e.g., FOAF\(^9\), DC\(^10\)) in order to avoid heterogeneity, there is great flexibility in terms of which vocabulary to use and which terms to use while describing data. Different publishers often express different, and even contradictory, views. Therefore, it is no surprise that such data sources often manifest a wide range of heterogeneities regarding structure and semantics.

The very nature of the WoD (where everyone can publish) is likely to cause several kinds of heterogeneity to exist at different levels mostly due to the lack or minor coordination of the publishers. However, it seems that there is an opportunity to leverage the semantic annotations that annotate RDF sources to help reconcile the heterogeneity that unavoidably arises.

As the WoD grows in popularity there is an increasing need for consumers to reconcile heterogeneity at different levels and query data from multiple datasets (that are likely to be independently and autonomously modified at any time). Considering the challenges, as well as the possible opportunities offered under the context of LD (e.g., deferenceable URIs, typed links, rich semantic annotations), this dissertation sets out to explore whether techniques proposed for pay-as-you-go data integration systems (seen as dataspaces) can be adapted for use over LD sources.

1.3 Dataspaces: a Pay-as-you-go Paradigm for Data Integration

Resolving the heterogeneities that data sources exhibit and providing a transparent single point of access to data that resides in multiple data sources has been the

\(^9\)http://xmlns.com/foaf/spec/
\(^10\)http://dublincore.org/documents/dcmi-terms/
focus of significant attention in the data management community. This challenge
was termed *data integration* [Lenzerini, 2002; Doan and Halevy, 2005; Halevy
et al., 2006b], with several proposals suggested as responses to this challenge.
For example, a *traditional* data integration system builds on the idea of virtual
integration (or mediator-based architecture [Wiederhold, 1992]), where an inte-
gration schema (called a *mediated schema*) is designed, based on the integration
requirements, and then exposed to the user. The mediated schema is purely a
logical schema used for the purpose of posing queries: the actual data still resides
in the data sources. Once a query has been posed over the mediated schema,
techniques for view-based query rewriting [Halevy, 2001] allow a user query to be
unfolded into queries that can be executed against the underlying data sources
that contribute to the integrated artefact. In order for the underlying sources to
interoperate, such systems are typically grounded on two basic capabilities: (a)
the ability to discover *semantic matches* at both the schema and instance levels
of the participating data sources so as to yield a set of semantic correspondences
(called *matches*); (b) to derive from a set of semantic correspondences, appro-
priate *semantic mappings*, which are executable expressions that transform the
data structured under the source schemas into a form that is compatible with the
integration schema.

Traditionally, data integration systems have required a significant up-front
effort to define and then maintain the semantic mappings between the integration
schema and the schemas of the underlying data sources before offering any services
to the user. Halevy [2005] stated that in a typical data integration scenario half of
the effort, up to 80% sometimes, is devoted to obtaining the semantic mappings
in a way that the semantics of the data are preserved. Thus, the deployment
of such a traditional data integration system is heavily dependent on expert-
intensive manual efforts during the matching and mapping stages, which makes
the deployment both resource- and time-consuming [Seligman et al., 2002; Halevy
et al., 2006b].

“*mappings are created by the builders or administrators of the sys-
tem, often in a laborious and error-prone process.*” McCann et al.
[2005].

As such, in spite of the high quality of integration outcomes that tradi-
tional data integration has to offer, such an approach has proved to be cost-
ineffective [Halevy et al., 2006a]. This has been the case particularly for large
scale deployments or for the integration of highly dynamic sources (e.g., Web resources) where the schemas and the data evolve over time very quickly. As a result of these dynamic changes, the quality of an integration is likely to decay rapidly, unless significant and constant effort is made into reflecting the changes through updated semantic mappings.

As a response to this state of affairs, the notion of an incremental *pay-as-you-go* style integration has emerged [Halevy, 2005; Franklin et al., 2005]. Simply put, rather than waiting for a high-quality resolution of the heterogeneities at the beginning, thereby “paying” a lot for an almost perfect integration of the sources, useful initial functionality is instead offered to the users, irrespective of how well integrated the sources are thereby stimulating users to provide feedback that leads to incremental improvement over time. This idea was initially proposed as a vision of *dataspaces* by Franklin et al. [2005].

> “the goal of dataspace support is to provide base functionality over all data sources, regardless of how integrated they are.” Franklin et al. [2005].

The “pay-as-you-go” philosophy relates the kind and amount of “effort” the users put into the system while using it, e.g., in the form of incremental user feedback (i.e., “as-you-go”) so as to improve the initial, potentially low-quality integration, e.g., in terms of the accuracy of query results [Belhajjame et al., 2013]. In a sense, and as stated by Halevy et al. [2006a]:

> “the dataspace approach postpones the labor-intensive aspects of data integration until they are absolutely needed.”

The manual intervention that is required to set-up a traditional data integration system is replaced with automated techniques to *bootstrap* the system [Madhavan et al., 2007; Sarma et al., 2008]. This involves techniques that automatically discover the matches and use these to derive semantic mappings between the mediated and the source schemas. Using such techniques, the system produces integrations that provide *best-effort* query/answering capabilities. The idea is that these suffice to motivate the users to adopt the integration without overly investing to set it up. Through usage, the hope is that the user interacts with the system by providing some sort of user feedback (e.g., [Belhajjame et al., 2011] and [Jeffery et al., 2008]), which eventually enables the incremental improvement
of the initial, speculative integration and brings the later closer to the user’s requirements. Over time, the idea is that the improvement (enabled by user feedback) compensates the initial lack of quality inherent in the extensive use of automation. As proposed by Hedeler et al. [2010b], a dataspace is expected to support the following features:

(i) minimal upfront initialisation costs with the use of automatic techniques.

(ii) support for incremental improvement by leveraging human feedback while interacting with the system.

(iii) support for the different levels of uncertainty that are inherent due to the use of automatic techniques for bootstrapping, and improving the derived integration.

To conclude, several researchers have envisioned the WoD as a web-scale dataspace [Heath and Bizer, 2011]. This analogy is one that this dissertation builds upon. LD enables various data sources from diverse domains to co-exist, and basic functionally in terms of retrieving information from the WoD is provided by following links, notwithstanding how integrated the sources are. In addition, there is an ongoing community-driven effort to mitigate the effects of heterogeneity at different levels. This evolutionary approach resembles the pay-as-you-go philosophy of dataspaces. Specifically, the more the publishers “pay” in terms of the effort invested in identifying heterogeneity in terminologies, resolving inconsistencies in values, identifying URIs that refer to the same entities at both the instance and conceptual-levels etc., the more homogeneous and integrated the ecosystem of the WoD will be. In this dissertation, we of course, do not claim to have addressed all the aforementioned problems. Specifically, this dissertation is concerned with mostly (i) and (iii), among the features highlighted by Hedeler et al. [2010b], while proposing techniques that can enable a pay-as-you-go data integration system (as realised by dataspaces) to be used for the integration of heterogeneous LD sources. The research contributions reported in this dissertation are now described in greater detail.

1.4 Aim, Objectives and Research Contributions

The aim of the research presented in this dissertation is to explore the extent to which LD sources can be treated, with respect to pay-as-you-go data integration,
as if they were classical structured sources, while leveraging the rich semantic knowledge offered by LD to reduce the uncertain decisions that inevitably arise.

To achieve the above aim, the following objectives are envisaged:

**O1** To identify the challenges and opportunities stemming from an attempt to apply pay-as-you-go data integration techniques to LD sources.

The research contribution resulting from this objective is an account of identified challenges and opportunities in LD integration when the principles and techniques used by pay-as-you-go data integration are applied when the latter is envisioned as a dataspace.

**Impact:** The results of the study of applying existing pay-as-you-go data integration techniques for the integration of LD sources were published in [Paton et al., 2012]. The purpose of this article was twofold: (a) to identify the challenges when existing techniques used in pay-as-you-go data integration (envisioned as a dataspace) are applied in a setting where the sources are LD sources; and (b) to enumerate the potential benefits from adopting the pay-as-you-go data integration principles for the integration of LD.

**O2** To devise techniques that overcome the problems that prevent LD sources from being treated as classical sources from the point of view of the dataspace life-cycle.

The research contribution resulting from this objective is a structure elicitation technique that uses a hierarchical agglomerative clustering to derive a structural summary of an RDF data source. The technique works over RDF triples, and uses a set of heuristics to elicit a structural summary of the source. The structure elicitation strategy is described in more detail in Chapter 2.

**Impact:** The developed technique was presented in [Christodoulou et al., 2013] and won the best paper award at the 3rd International Workshop on Linked Web Data Management held in conjunction with EDBT 2013. An extended version of this work [Christodoulou et al., 2014] has been accepted for publication in LNCS Transactions on Large Scale Data and Knowledge Centered Systems (TLDKS) for the special issue on Big Data and Linked Open Data.
To develop techniques that take advantage of the availability of the rich semantic annotations used to describe LD sources in order to improve the outcome of the bootstrapping phase in the dataspace life-cycle.

The research contributions stemming from this objective are as follows:

- A principled methodology for reasoning under uncertain situations that are likely to arise due to the automatic techniques used to bootstrap a dataspace management system.
- A set of experiments for initialising the approach through the construction of probabilistic models for similarity scores returned by different schema matching techniques, and semantic annotations from LD vocabularies.
- An evaluation of the developed techniques and methodologies when these are applied to the assimilation of different pieces of evidence to assign degrees of belief on hypothesised construct equivalence.

1.5 Overview of Thesis Structure

The remainder of this thesis is structured as follows:

- Chapter 2 begins with an introduction to the main concepts and Semantic Web technologies that support the LD paradigm. The several challenges that arise in consuming data in a WoD setting along with a study of proposals that relate to the integration of data from multiple RDF data sources is then provided. This is followed by a discussion of the dataspace life-cycle and on how dataspaces enable incremental integration on demand. A case-study (using real LD datasets) is then discussed illustrating how pay-as-you-go data integration techniques can be adopted for the integration of multiple, possibly heterogeneous RDF data sources. The chapter concludes with a discussion of uncertainty management in dataspaces.

- Chapter 3 describes a contribution to the problem of eliciting structures from LD sources. In the LD context, publishers can choose terms from a variety of vocabularies to annotate their RDF data. The RDF model does not require a definition of a schema (i.e., conceptual description) of the RDF data source to be available before access to the data. In LD,
this classical requirement of traditional databases is relaxed. The lack of a conceptual description requires prior inspection of the data sources before they can be consumed. For database-inspired data integration, schemas are important both for reconciling the semantic heterogeneities that the sources may exhibit and for defining the mappings that translate data from the sources into a form that is compatible with the mediated schema. Chapter 3 proposes a methodology for inferring such conceptual descriptions directly from LD datasets.

• Chapter 4 argues that the semantic knowledge from LD vocabularies that is used to annotate RDF data can have an impact on improving uncertain decisions made by the automatic matching techniques typically used to bootstrap a dataspace. The chapter starts by discussing how Bayes’s theorem can be used to dynamically increase or decrease a prior degree of belief in the light of evidence. The chapter then elaborates on Bayesian updating and demonstrates how one can reason on a hypothesis given the evidence of similarity scores from string-based matchers on one hand and of semantic annotations from LD vocabularies on the other. The experimental studies carried out to evaluate this approach show that: (i) the proposed Bayesian methodology for combining evidence from different heterogeneous matching systems (instead of an average) in all cases delivers more certain judgements on construct equivalence decisions; and (ii) the existence of semantic annotations can be leveraged as evidence to improve the decision making of matching techniques that typically base their results on syntactic evidence alone.

• Chapter 5 summarises the resulted contributions and discusses on their significance. In addition, opportunities for future research directions are pointed out along with a discussion on some limitations of the contributions described.

We call the attention of the reader that the dissertation is structured in such a way that technical background appears in Chapter 2 but that the discussion of related work is distributed across each contribution chapter (i.e., Chapters 3 and 4) where they can be discussed in more focus.
Chapter 2

Anatomy of Dataspaces over Linked Data

“In the middle of difficulty lies opportunity.”

Albert Einstein

The vision of LD is that of a paradigm that builds on several Semantic Web technologies to publish, interlink and share information on the Web in a format that is both useful for humans and for software agents. With respect to publication, LD follows the traditional Web of documents in providing low barriers to entry in order to encourage a diversity of publishers to make their datasets publicly available on the Web. As regards consumption, crawlers have been developed that can index the WoD, allowing consumers to search and browse the data by following links and to discover data in a follow-your-nose fashion. Indeed, this simple practice provides benefits. However, what is really consumed by users on the WoD is essentially what has been published, i.e., there is little systems-added value. Therefore, if publishers follow best publication practices [Hyland et al., 2014] (e.g., avoiding heterogeneity issues through the reuse of terminology from widely-deployed vocabularies, providing external links to other LD datasets so that more knowledge can be discovered and maintaining such links), then a WoD resource can become useful for both humans and software agents. This chapter argues that there is more value to be added for consumers of the WoD by providing a coherent and integrated view over multiple LD datasets.

In Chapter 1, we briefly reviewed the heterogeneity issues that LD sources experience in terms of: (i) different URIs representing the same real-world entities
or abstract concepts; (ii) contradictions in predicate values; (iii) format inconsistencies in value representations; (iv) inconsistent structural representation for concepts; and (v) terminology.

Providing consistent access to semantically-integrated data requires that different aspects of heterogeneity, as is the case with those mentioned, are reconciled at web scale. Traditional data integration has proven to be cost-ineffective for deployment at Web scale [Madhavan et al., 2007]. As a response to this challenge, the idea of pay-as-you-go data integration was proposed under the vision of dataspaces. Section 1.3 argued that techniques from dataspaces, such as the automatic inference of semantic correspondences or the derivation of semantic mappings, and principles, such as payment in the form of user feedback, can potentially help with respect to the publication and consumption on the WoD. Following on from there, Section 2.6 discusses the dataspace life-cycle [Hedeler et al., 2009] (i.e., bootstrapping, usage and improvement) and illustrates (with a running example) how techniques from pay-as-you-go data integration can be adopted for the integration of LD datasets.

This chapter begins with an introduction to the fundamentals and main concepts behind LD principles and Semantic Web technologies that make the paradigm of publishing and interlinking structured data on the Web possible.

2.1 The Semantic Web Vision

The World Wide Web (or simply the Web) as proposed by Berners-Lee and Cailliau [1990] has served its purpose for many years as a global medium for sharing, linking and exchanging information, mostly in the form of Web documents accessed remotely via the Internet. Documents on the Web are identified by a unique global address, a Uniform Resource Locator (URL). URLs can give access to resources using the universally available HyperText Transfer Protocol (HTTP). Traditionally, one can retrieve a Web document by following its HTTP URL in a Web browser, i.e., an application that enables navigation on the Web and can render the information encoded in the document. Information in a Web document can take several forms, such as formatted natural language or images, and also includes rendering instructions, all defined by means of the HyperText Markup Language (HTML). A crucial characteristic of the Web of documents is the fact that documents have embedded URLs that link them to related Web documents
(potentially residing on different Web servers). Such hyperlinks enable navigation, and rely on simple notions [Jacobs and Walsh, 2004], such as HTTP URLs and HTML. This simplicity characterises the first era of the Web as a common medium for exchanging knowledge represented as hyperlinked documents.

On the Web of documents, humans are the primary actors in terms of understanding the meaning of Web content and drawing conclusions from the available information. Despite the various advances in the area of natural language processing and machine learning techniques, machines (or software agents) cannot effectively make significant use of the plethora of heterogeneous knowledge sources available on the Web, or act upon the available information. This is in part due to the lack of a uniform format for structuring and publishing machine-readable knowledge, and of explicit semantics for disambiguating its meaning. The Semantic Web envisions an ecosystem where machines are empowered by the ability of processing Web content, and are able to draw conclusions from the information they process.

“the Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.” [Berners-Lee et al., 2001]

To facilitate the realisation of the Semantic Web (SW) vision, the World Wide Web Consortium (W3C) have published several Web standards and technologies. These include a common data model (i.e., Resource Description Framework (RDF) [Manola and Miller, 2004]) for representing machine-readable data in a structured form, syntax that allows machines to parse formal text (e.g., XML, Turtle\textsuperscript{1}), formal languages to define the semantics of data such as RDF Schema (RDFS) [Brickley and Guha, 2004] and Web Ontology Language (OWL) [Mcguinness and van Harmelen, 2004], as well as ways to declaratively query the structured content on the Web using SPARQL [Prud’hommeaux and Seaborne, 2008].

The ultimate goal of the SW is to make much of the Web content available in a machine-understandable format, so that knowledge can be shared, reused, integrated and interpreted in such a way that machines can draw conclusions from it. The recent paradigm of linked data is a pragmatic approach towards achieving the SW vision. Subsequent sections present the SW technologies that come

\textsuperscript{1}Terse RDF Triple Language [Becket and Berners-Lee, 2008] is inspired by the Notation3 syntax.
together under the LD paradigm to facilitate the publication and interlinking of structured data on the Web.

2.2 The Linked Data Paradigm

The Linked Data (LD) paradigm has evolved as a pragmatic approach for the transition from a document-oriented Web to a Web of structured and interlinked data, which will ultimately bring about one aspect, at least, of the Semantic Web vision [Auer et al., 2013]. In essence, the idea of LD relates to a set of best practices for the publication and interlinking of structured data on the Web [Bizer et al., 2009a]. These best practices were outlined by Berners-Lee [2006] and are known as the linked data principles. The principles provide guidelines on the usage of Web technologies and standards, initially developed for the SW, to enable the publication of structured data on the Web in such a way that enables data to connect by setting data-level links from different sources. This practice of setting links comes by analogy with the classic Web of documents where “anything can link to anything” with the use of hyperlinks. The principles, claim that connecting data with the use of data-links develops the Web into a global data space where data co-exists; i.e., a Web of (Linked) Data. The four principles are as follows (paraphrasing [Berners-Lee, 2006]):

(LDP-1) use URIs as names to things;

(LDP-2) use HTTP URIs so that the things can be looked up (i.e., dereferencing);

(LDP-3) provide useful information upon lookup using the standards (RDF, SPARQL);

(LDP-4) include links by using URIs that refer to related RDF documents and allow further navigation.

Technologically, the paradigm underlying the LD principles builds upon the use of global identifiers not only for the identification of Web documents, but also for the identification of real-world entities (i.e., objects or abstract concepts). The HTTP protocol acts as a universal mechanism for accessing the URIs assigned to entities; it is responsible for resolving the URIs to the locations where structured
data that describe the entity can be found and returned (called dereferencing).

LD principles advocate that data be represented using a single model in the form of subject-predicate-object RDF triples (see Section 2.2.3). Linking between data items takes the form of RDF links where the subject URI of a triple belongs to one namespace\(^2\) and the object is a URI in the same, or in some other namespace (see Section 2.2.4). RDF represents data in a form that makes them self-descriptive, where links point to the actual definitions of the terms used to describe the data from various LD vocabularies defined in SW languages (see Section 2.3).

Since the publication of the LD principles, an aspect to the Web began to emerge, viz., one that builds on data made available under a structured and machine-understandable format. A community effort was established under the Linking Open Data project to bootstrap the WoD with open licence data sources by adopting the LD principles. Specifically, the goal of the project was twofold: (i) to identify, re-publish and interlink open license data sources as RDF on the Web; and (ii) to highlight the opportunities and benefits of LD to the broader community [Bizer et al., 2009a].

One of the main results of this effort is the LOD cloud, containing various interlinked RDF data sources from different domains. The most prominent example of which is the DBpedia, which describes in RDF more than 2.6 million entities extracted from semi-structured Wikipedia articles [Bizer et al., 2009b]. Since its inception, many publishers and third parties have joined the campaign to publish structured data on the Web as LD, with data sources spreading over a variety of domains e.g., geographic information, academia, music, governmental data, biomedical data, etc. By September 2011, it was estimated that 295 sources fulfilling the basic LD principles\(^3\) had been added to the LOD cloud, comprising of 31.6 billion triples. With the popularity of LD gradually growing and with more publishers adopting them, by July 2014\(^4\) the WoD consisted of 2,122 datasets, with over 61.9 billion triples asserted.

This resulted in a growing and varied collection of heterogeneous data sources, where:

- data are interlinked with other structured data on the Web,
- entities are identified and discovered with the use of Web addresses,

\(^{2}\)a common URI prefix scheme.

\(^{3}\)http://lod-cloud.net/state/

\(^{4}\)http://stats.lod2.eu/
data are semantically annotated, and

- where navigation in a *follow-your-nose* fashion with the use of links allows the discovery of new information.

The WoD can be seen as being closely inspired by the existing Web and shares many of the properties of the latter [Bizer, 2009] while aiming to move closer to the vision of a SW, as follows:

- The WoD offers low barriers to entry. Anyone can publish their data on the WoD as long as they conform to the minimum requirement of using RDF as a unified model for representing them.

- Links connect entities, creating a global data graph on the Web that enables crawling, navigation and the discovery of new information.

- Data are annotated with semantics that make them self-describing. URIs point to definitions of the vocabulary terms used to annotate the data.

- The WoD is an open environment that enables applications to discover new sources of information on the fly, using the links between them.

However, the realisation of the WoD as a global, heterogeneous database that builds on SW technologies faces its own challenges, and gives rise to opportunities and potential new application categories. Bizer et al. [2009a] envision the following:

“This Web of Data enables new types of applications. There are generic Linked Data browsers which allow users to start browsing in one data source and then navigate along links into related data sources. There are Linked Data search engines that crawl the Web of Data by following links between data sources and provide expressive query capabilities over aggregated data, similar to how a local database is queried today. […] Unlike Web 2.0 mashups which work against a fixed set of data sources, Linked Data applications operate on top of an unbound, global data space. This enables them to deliver more complete answers as new data sources appear on the Web.” [Bizer et al., 2009a]
This dissertation shares the view that there is a need for data integration on
the Web that will enable structured query answering, not just keyword search-
ing, over aggregated data. As we shall discuss in subsequent chapters, we pro-
pose techniques that contribute to the bootstrapping stage of a pay-as-you-go
approach, adapted for the integration of LD sources. The following sections will
present the details of the SW technologies and standards that comprise the LD
paradigm.

To support our discussions, and to explain the various concepts and technolo-
gies, we use a running example. Figure 2.1 shows an RDF (sub-)graph retrieved
by following the HTTP URI that was assigned to describe data about a musi-
cian http://dbtune.org/magnatune/artist/mijo, which is made available by
the Magnatune RDF data source\textsuperscript{5}. Prefixes for the abbreviated CURIE\textsuperscript{6} names
are used in the following section, and throughout the dissertation, and can be
found in Appendix G. The rounded boxes with dark border lines illustrate RDF
instance level data that is retrieved by dereferencing the HTTP URI assigned to
the musician mja:mijo.

In contrast to the RDF graphs retrieved at the instance level, the rounded
boxes with dotted lines show definitions of terms used to annotate the instance
data at the vocabulary-level (what we refer as the conceptual level). More specif-
ically, the dotted boxes illustrate a subset of the Music ontology\textsuperscript{7}, the FOAF
ontology\textsuperscript{8}, the DBpedia ontology\textsuperscript{9} and terms from Schema.org\textsuperscript{10}. The definitions
of the vocabularies are in RDFS or OWL, although some of the predicates and
values used are missing from the RDF sub-graphs to make them more readable.
For a complete definition of the RDF graphs made available at the instance level
and the LD vocabulary definitions, the interested reader may wish to retrieve the
actual definitions as published on the WoD by following the URI links provided
though this is not essential for understanding and appraising the contributions
reported.

\begin{itemize}
\item \textsuperscript{5}http://dbtune.org/magnatune/; retr. 2014/01/04
\item \textsuperscript{6}W3C, “CURIE Syntax 1.0: A syntax for expressing Compact URIs”, http://www.w3.org/TR/curie, 2010.
\item \textsuperscript{7}http://purl.org/ontology/mo/
\item \textsuperscript{8}http://xmlns.com/foaf/0.1/
\item \textsuperscript{9}http://dbpedia.org/ontology/
\item \textsuperscript{10}http://schema.org/
\end{itemize}
Figure 2.1: RDF sub-graph from Magnatune with sample annotations at the conceptual level.
2.2.1 Resource Identification with URIs

The first LD principle, **LDP-1**, advocates the use of URIs as the identification mechanism not only for Web documents, but also for the identification of *things* (e.g., real-world entities or abstract concepts). Instead of using application specific identification for things e.g., database keys, the principle suggests a generic universal identification mechanism by using on-line resources\(^{11}\). More specifically, things are identified using HTTP URIs so that they can be retrieved using the HTTP protocol – a universal mechanism for retrieving resources on the Web. Following our example in Figure 2.1, the HTTP URI `http://dbtune.org/magnatune/artist/mijo` is assigned to describe a German musician called Mijo (a.k.a Michel Jordan). LD uses URIs mainly for two reasons: (i) as a generic means to uniquely assign names to things in a decentralised fashion; and (ii) as a uniform way to access distributed information used to describe the identified entity.

2.2.2 Dereferenceable URIs

As per **LDP-2**, URIs should be dereferenceable using the HTTP protocol so that links to things can be looked up to obtain a description of the identified real-world entity or abstract concept. Typically, information on the Web is returned in the form of Web HTML documents. In the context of LD, URIs are used to identify things where descriptions for the identified things are represented in a machine-readable format (i.e., as RDF data). As previously mentioned, the Web is a medium used to share information which is intended to be useful for both actors i.e., humans and machines. It is therefore critical that both actors should request and retrieve information in a format that is understandable to them. On the Web, this can be achieved using a content negotiation mechanism [Fielding et al., 1999], which is used to disambiguate the type of Web documents returned i.e., HTML for human, or RDF for machine consumption.

\(^{11}\)LD builds on the Web architecture [Jacobs and Walsh, 2004], the technical term *resource* is used to refer to *things of interest*, in this dissertation we may use this term as well.
2.2.3 A Uniform Data Model: RDF

LDP-3 advocates the use of a uniform data model for publishing structured data on the Web; this requirement is met by the Resource Description Framework (RDF). The use of an agreed-upon data model by independent publishers is important to ensure the interoperability of data notwithstanding the domain or procedure used to generate them. RDF is a language initially developed for the SW with a view to annotating data with machine-readable meta data [Lassila and Swick, 1999]. In the context of LD, RDF is used to represent structured information about resources on the Web [Manola and Miller, 2004]. Note that a resource can be anything to which a URI can be assigned (either a Web resource, a real-world entity or an abstract concept). This section discusses RDF as a basic data model through which information is described by making statements about resources, whereas Section 2.3 discusses RDF Schema [Brickley and Guha, 2004], which extends RDF with basic ontological modelling constructs.

In RDF, information is represented with a simple structure in terms of statements consisting of a subject, predicate and object, referred to as RDF triples. The subject and predicate positions in an RDF triple are URIs that identify any resource, whereas the object position can be either a URI, or some data value represented as a literal. In addition, RDF allows the subject and object positions to use a special kind of local identifier called a blank node, which can be only referenced locally. This dissertation adopts the formal semantics of RDF as described in [Hayes and McBride, 2004].

**Definition 1 (RDF Term, Triple).** Given a set of URI references $\mathcal{U}$, a set of blank nodes $\mathcal{B}$, and the set of both plain and datatype literals $\mathcal{L}$, where $\mathcal{U}$, $\mathcal{B}$ and $\mathcal{L}$ are pairwise disjoint, the elements $e \in (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ are called RDF terms. A triple $t = (s, p, o) \in (\mathcal{UB} \times \mathcal{U} \times \mathcal{UBL})$ (where, e.g., $\mathcal{UB}$ is a shortcut for $\mathcal{U} \cup \mathcal{B}$) is called an RDF triple. Here, $s$ represents the subject, $p$ the predicate, and $o$ the object of the RDF triple.

The object in a triple can take the role of subject in another triple giving rise to a directed labelled graph where subjects and objects correspond to the nodes of the graph and predicates correspond to the edges (see Figure 2.1). More formally:

**Definition 2 (RDF Graph).** An RDF graph is a finite set $G$ of RDF triples such that $G \subset (\mathcal{UB} \times \mathcal{U} \times \mathcal{UBL})$. In this dissertation, we sometimes refer to an RDF graph as an LD dataset, or an RDF (data) source, or a LD (data) source.
We now introduce some of the terminology and features of RDF that are relevant throughout the dissertation. We use the Turtle syntax [Beck et al. and Berners-Lee, 2008] instead of the official W3C standard for serialising RDF in XML (called RDF/XML [Beckett and McBride, 2004]) because it is much easier to read and write.

**RDF Terms**

An RDF triple draws component values from three disjoint sets $U$, $B$ and $L$. The set $U$ is the set of URIs that are RDF terms. For example, http://dbtune.org/magnatune/artist/mijo is a URI assigned to a German musician. Sometimes it is convenient to abbreviate URIs to a shorter form by the use of CURIE-style abbreviations. Turtle allows the use of CURIE shortcuts by defining a reusable prefix e.g., @prefix mga: <http://dbtune.org/magnatune/artist/>.

With the definition of a prefix, the full URI for Mijo can be abbreviated using the prefix:localname format e.g., mga:mijo, as shown in Figure 2.1.

The set $L$, is the set of literals that are RDF terms. Data values in RDF take the form of a sequence of characters (i.e., strings) which may be of two types, plain or typed. Plain literals are usually enclosed in double quotes, e.g., "Michel Jordan", potentially with a tag that shows the language of the string, such as "Michel Jordan"@en. In addition, literals can be typed by carrying an additional datatype identifier. For example, in Figure 2.1, the track with URI mgt:3956 was created on "2005-09-05"^^xsd:date. Datatypes are identified using URIs, where many of the datatypes in RDF are the ones defined in XML Schema [Biron et al., 2004], such as xsd:date. When a literal is typed, the meaning of the string is determined by the datatype. In our example, the string "2005-09-05" is not just a sequence of characters, but represents a date. In cases where literals are untyped, they are interpreted as being implicitly typed as xsd:string. When drawing RDF graphs, literals are represented with rectangles to distinguish them from nodes that refer to URI resources (see Figure 2.1).

In addition to URIs of resources and literals, RDF allows resources to be anonymous, meaning that no globally unique URIs have been defined for them. Resources that do not explicitly have a URI are known as blank nodes (or bnodes), and can only be referenced locally within the RDF document in which they are defined [Hayes and McBride, 2004; Mallea et al., 2011]. Frequently, blank nodes are used to model multi-valued relationships, or to capture composite information.
about a resource that is to be known locally. However, as discussed by Heath and Bizer [2011], the use of blank nodes is best avoided in the context of LD. In Turtle, blank nodes are denoted by the prefix _: or by using square brackets [ ]. Local identifiers for resources comprise the set B.

### RDF Triples, Quadruples and Graphs

As mentioned previously, information in RDF is structured by making statements about things using RDF triples. For example, in the statement “Mijo is based in Germany”, “Mijo” is the subject, “based in” is the predicate (or property) and “Germany” is the value of the predicate (the object). Following our example from Figure 2.1, this statement is expressed with the RDF triple in Listing 2.1, where mga:mijo is the subject, foaf:based_near is the predicate, and dbr:Germany is the value of the object using CURIE-style abbreviations.

```
# PREFIX DECLARATIONS
@prefix mga: <http://dbtune.org/magnatune/artist/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

# RDF Triple
mga:mijo foaf:based_near dbr:Germany .
```

Listing 2.1: Example of an RDF triple in Turtle syntax.

In this case, the value of the object is the subject URI of some other triple that provides information about Germany, giving rise to an RDF graph, as in Listing 2.2. Note that RDF triples using Turtle are delimited using a trailing ‘.’ and triples that have the same subject URI are grouped using ‘;’. In RDF, a triple that consists of three URI resources, where the identifiers at the subject and object positions are from different namespaces, is called an RDF link. The URI at the predicate position determines the type of the relationship. Information on the WoD is decentralised and RDF links are the glue that connect resources from different data sources into a global data space. The RDF data provided in Figure 2.1 shows an example RDF (sub-)graph from the WoD that illustrates how it is possible to discover and combine information by following links from RDF graphs that originate from different namespaces. One can follow the link foaf:based_near to retrieve more information about Germany. As we shall subsequently discuss, the fourth LD principle suggests the use of external links
into other data sources on the WoD, thus enabling the discovery of new knowledge in a *follow-your-nose* fashion [Heath and Bizer, 2011]. Section 2.2.4 discusses in more detail the types of RDF links.

```
# PREFIX DECLARATIONS
@prefix mga: <http://dbtune.org/magnatune/artist/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

# RDF Triples
mga:mijo foaf:based_near dbr:Germany ;
  foaf:name "Michel Jordan"@en .

dbr:Germany dbo:capital dbpedia:Berlin ;
  dbo:areaTotal "3.57e+11"^^xsd:double .
```

Listing 2.2: Example of an RDF link in Turtle syntax.

A *set* of RDF triples (as in Listing 2.2) is formally referred to as an *RDF graph* (as per Definition 2). Being a set means that the triples in an RDF graph can have an arbitrary order, and that duplicate triples are not allowed [Manola and Miller, 2004]. For the purposes of completeness, Table 2.1 summarises the anatomy of an RDF Statement.

<table>
<thead>
<tr>
<th>Statement</th>
<th>RDF Term</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject</td>
<td>either a URI or bnode</td>
<td>resource being described by the RDF triple</td>
</tr>
<tr>
<td>predicate</td>
<td>must be a URI</td>
<td>states the <em>relationship</em> between the subject and the object</td>
</tr>
<tr>
<td>object</td>
<td>can be a URI, bnode or literal</td>
<td>holds the <em>value</em> of the relationship</td>
</tr>
</tbody>
</table>

Table 2.1: Anatomy of an RDF statement.

Sometimes it may be required to associate to an RDF triple information that captures its context or provenance. This gives rise to a pair that is referred to as an *RDF Quadruple* [Cyganiak et al., 2008]. For example, an RDF triple, e.g., Listing 2.1, can be annotated with provenance information indicating that it is defined under the RDF document identified by the URI `http://dbtune.org/magnatune`. 
Definition 3 (RDF Quadruple). An ordered pair \((t, c)\), where \(t\) is an RDF triple \(t := (s, p, o)\) and \(c\) is the context of \(t\). We sometimes refer to an RDF Quadruple as an N-Quad or simply as a quadruple.

Note that an RDF graph is itself a resource; thereby it can be represented by a URI, the graph URI. Listing 2.3 shows an example of an RDF Quadruple with the graph URI as the context.

```turtle
# PREFIX DECLARATIONS
@prefix mga: <http://dbtune.org/magnatune/artist/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

# RDF Quadruple
```

Listing 2.3: Example of an RDF Quadruple in Turtle syntax.

RDF Serialisations

The W3C standard for serialising RDF is the RDF/XML syntax, or else RDFa [Adida and Birbeck, 2008] for embedding RDF triples into HTML documents. However, several alternative serialisation formats for publishing RDF graphs on the Web have been developed [Auer et al., 2013]. As previously mentioned, in this dissertation we use the more readable Turtle notation, where RDF triples are separated using a trailing '.', URIs are denoted by brackets <> and double quotes "" represent literals. Furthermore, subsequent chapters may use the ‘a’ Turtle shortcut for rdf:type, which is used in RDF graphs to denote class membership. When convenient, we use visual representations of RDF graphs (e.g., Figure 2.1). Turtle syntax is also used for formulating SPARQL queries, as discussed below.

2.2.4 Include Links to Other URIs

The WoD becomes an interlinked space of data through the use of links, as advocated in LDP-4. The last LD principle advocates the use of links to external data sources on the Web to enable the discovery of new knowledge through navigation and dereferencing:
"[. . . ] after you have published your information as Linked Data, you should ensure that there are external RDF links pointing at URIs from your dataset, so that RDF browsers and crawlers can find your data." [Bizer et al., 2007]

RDF links to external data sources are, typically, RDF triples, whose subject and object terms are URIs from different namespaces. Dereferencing the object URI from one namespace allows further navigation to a different data source which is stored in another remote server. Heath and Bizer [2011] distinguish between three types of RDF links on the WoD:

(i) relationship links – links that point to additional knowledge that relates to some subject URI that enables the discovery of new knowledge. For example, retrieving the object URI of the predicate foaf:based_near yields the discovery of more knowledge about the location of the musician with URI mga:mijo;

(ii) identity links – links that relate URIs to other URIs that identify the same real-world entity or abstract concept. Equivalence between resources at the instance level are typically expressed as owl:sameAs links, or with other similar terms to indicate that resources are somehow related e.g., skos:related [Nunes et al., 2013]. An example of an identity link is shown in the RDF triple <dbr:Germany owl:sameAs freebase:Germany>, indicating that freebase:Germany is an alternative resource that describes Germany (see Figure 2.1); and

(iii) vocabulary links – these point to the definition of vocabulary terms that are used to semantically annotate the data. For example, the predicate with URI foaf:firstName points to the definition of the predicate defined in the FOAF vocabulary (more details on LD vocabularies can be found in Section 2.3).

Additionally, vocabulary links are used to deal with various heterogeneities that might exist between the different vocabularies published on the WoD (more in Section 2.3). Such RDF links express mappings at the conceptual level, i.e., using RDFS and OWL terms, as typed links, e.g., owl:equivalentClass and owl:equivalentProperty to state that terms are equivalent.
Alternatively, if less strict mappings are appropriate, then these are expressed using, e.g., `rdfs:subClassOf`, `rdfs:subPropertyOf`, `skos:broadMatch`, and `skos:narrowMatch`. For example, as shown in Figure 2.1, `dbo:Country` is equivalent to `schema:Country`. Meta-information in the form of mappings enables LD applications to translate data into a representation they understand (e.g., into the application’s target schema) in a follow-your-nose fashion [Heath and Bizer, 2011].

In September 2011, a statistical analysis of the state of the LOD cloud [Bizer et al., 2011] found that, of the 31 billion RDF triples published, fewer than 500 million represented links between other RDF data sources. A more recent survey by Ermilov et al. [2013] showed that, of about 12.2 billion RDF triples analysed, only 0.8% were `owl:sameAs` links. The statistical analysis showed that, in general, the existence of links on the WoD remain very sparse. Despite the fact that publishers made their datasets available as RDF graphs, it seems that there is a cost in publishing links between datasets that the publishers are not willing to pay. Chapter 4 describes a methodology where mappings (expressed as RDF links) between LD vocabularies are leveraged by a Bayesian inference technique to make judgements on construct equivalence.

### 2.3 Vocabularies, RDF Schema and OWL

Thus far, previous sections have discussed how data can be structured as RDF triples and published on the Web. RDF allows one to assert propositions about resources that are identified by URIs. However, nothing has been discussed as regards the domain-specific terms that are used to describe the relationships between resources, nor the semantics that are used to annotate resources, nor how such resources can be organised into classes that share certain features. As regards the last point, the core RDF vocabulary made available under the URI [http://www.w3.org/1999/02/22-rdf-syntax-ns#](http://www.w3.org/1999/02/22-rdf-syntax-ns#) provides a built-in term to express class memberships. In particular, the term `rdf:type` is used (optionally) to assert that a particular resource is an instance (i.e., member) of some class.

To provide an example, in Figure 2.1, the resource `mga:mijo` is annotated with `rdf:type` to assert the fact that the resource is an instance of `mo:MusicArtist`, i.e., the class of musicians. At the conceptual (a schema or terminological) level, the class of musicians is also identified with a URI and is, in turn, defined as

12URI terms that appear in the *predicate* position of an RDF triple.
an instance of the `rdfs:Class`, denoting the class of all classes. As discussed later, the semantics of classes e.g., `mo:MusicArtist` can be defined using SW ontology languages (i.e., RDFS or OWL). In RDF, it is possible that resources share common features with many classes and that classes have many resources as instances.

In addition, the RDF vocabulary provides means to define terms that appear in the predicate position of an RDF triple. Such terms are defined as being instances of the built-in `rdf:Property` class, denoting the set of all properties. For example, the predicate `foaf:firstName` in Figure 2.1 is defined as a property in some LD vocabulary (e.g., FOAF). This section does not cover all the terms in the RDF vocabulary, nor does it elaborate on the model theoretical semantics of RDF; thus, for a comprehensive description of the formal RDF semantics, the reader is referred to [Hayes and Patel-Schneider, 2014]. It is however, important to briefly elaborate on the notion of entailments (i.e., the valid conclusions) which, when given an RDF graph\textsuperscript{13}, formalises which other statements are entailed by, i.e., are logical consequences of, the statements comprising the graph [Hayes and Patel-Schneider, 2014]. In the SW, this formalisation lays the foundations for machine-readable knowledge.

In LD, a collection of property terms used as predicates to describe resources (e.g., `mga:mijo`), their relationships with other resources (e.g., `foaf:based_near`) and classes (e.g., `mo:MusicArtist`) is usually called a vocabulary (what we sometimes call a LD vocabulary). In reality, vocabularies are collections of URI identifiers with a clearly defined meaning [Heath and Bizer, 2011] described using SW ontology languages, either the more basic RDF Schema (RDFS) language or the more expressive Web Ontology Language (OWL). The semantics of classes and properties are defined claims, as are the entailments, that each language is able to express when realised as inference rules [ter Horst, 2005]. Inference rules match parts of a given RDF graph to derive conclusions as a consequence of the available knowledge encoded as RDF triples. A subset of inference rules from RDFS and OWL [Carroll et al., 2012] is used in Chapter 4. This section briefly overviews some of the important features of RDFS and OWL in the LD context.

\textsuperscript{13}RDF semantics assume the Open World Assumption.
RDF Schema

RDF Schema is a lightweight ontology language that was designed to add more expressiveness to the RDF vocabulary. Although the core RDF vocabulary itself provides the means for expressing class membership using the \texttt{rdf:type}, it lacks expressivity in terms of defining new classes (i.e., new members of the RDFS meta-class \texttt{rdfs:Class}) or of modelling class and property hierarchies. RDFS provides a vocabulary with predefined semantics to express class and property hierarchies using the RDF properties \texttt{rdfs:subClassOf} and \texttt{rdfs:subPropertyOf} respectively (see Figure 2.1 for an example). In addition, RDFS provides the means to associate a class to the subject and object of a property using the terms \texttt{rdfs:domain} and \texttt{rdfs:range}. In LD, the above RDFS features allow one to model classes, binary relations between classes, class and property hierarchies, thus describing and embedding resource semantics in an RDF graph.

The Web Ontology Language

OWL is a more expressive SW ontology language. It extends RDFS to provide more expressive semantics and inference rules. OWL can capture complex conceptual (i.e., schema) relationships. For example, OWL allows one to express equality over instances (or individuals) using \texttt{owl:sameAs}, equivalence of classes (\texttt{owl:equivalentClass}) or properties (\texttt{owl:equivalentProperty}), disjointness of classes (\texttt{owl:disjointWith}) or properties (\texttt{owl:propertyDisjointWith}). OWL is built on top of RDFS in that it reuses the core RDFS vocabulary, while extending it to allow more complex class definitions and complex relationships. A full account of the semantics, features, OWL profiles [Motik et al., 2012; Krötzsch, 2012] (i.e., a set of sub-languages that allow different expressivity capabilities) and inference rules allowed in OWL is beyond the scope of this dissertation. The interested reader is referred to [Carroll et al., 2012].

In the context of LD, subsets of the RDFS and OWL vocabularies are used to provide domain-specific definitions of classes and properties with formal semantics. This dissertation refers to RDF graphs that structure terminological data with RDF triples as being at the conceptual level, thereby distinguishing them from RDF graphs that use such terminological data from vocabularies to describe the data or annotate resources as instances of classes, which we refer to the instance level. The formal definitions needed are as follows:
Definition 4 (Class). Given an RDF triple $t = (s, p, o)$, we refer to a class as an RDF term that appears either in: the $s$ position of $t$, where $p$ is rdf:type and $o$ is rdfs:Class or owl:Class; or in the $o$ position of $t$, where $p$ is rdf:type.

Definition 5 (Property). Given an RDF triple $t = (s, p, o)$, we refer to a property as an RDF term that appears either in the $s$ position of $t$, where $p$ is rdf:type and $o$ is rdf:Property or its subproperties owl:ObjectProperty or owl:DatatypeProperty; or in the $p$ position of $t$.

We frequently refer to both classes or properties, without explicitly distinguishing them, by the more generic term constructs.

Definition 6 (Meta-class). A meta-class is a class whose members are themselves either classes or properties. These are classes defined in RDFS and OWL specifications, where examples include rdfs:Class, rdf:Property, owl:Class, owl:ObjectProperty, owl:DatatypeProperty, owl:TransitiveProperty, owl:AnnotationProperty, etc.

Definition 7 (Meta-property). A meta-property is a property that has a meta-class as its domain. These are properties defined in RDFS and OWL specifications, where examples include rdfs:domain, rdfs:subClassOf, rdfs:subPropertyOf, owl:equivalentClass, owl:equivalentProperty, owl:disjointWith, owl:propertyDisjointWith, etc.

2.4 SPARQL: Querying RDF

This section gives an overview of the SPARQL query language for RDF [Prud-Hommeaux et al., 2008; Harris and Seaborne, 2013] (pronounced as sparkle) to the extent required by subsequent chapters. Apart from being the recommended query language for RDF graphs, SPARQL is also a protocol for communicating queries between a client and a server [Feigenbaum et al., 2013]. Here, we focus on SPARQL as a query language for RDF, following the semantics in [Pérez et al., 2009] and, for SPARQL 1.1, in [Aranda et al., 2013]. However, to conform with the preliminary discussions on RDF in Section 2.2.3, and for convenience, we focus on URIs instead of IRIs\(^\text{14}\).

\(^{14}\)An IRI Internationalized Resource Identifier [Dürst and Suignard, 2005], is an identifier of resources which extends the syntax of URIs with more characters for internationalization purposes.
SPARQL is a query language over RDF graphs where the query itself is formulated as a simple RDF graph comprising of a set of query patterns represented in Turtle syntax. In addition, query variables, which act as wild-cards, are specified within a query pattern to stipulate the expected result of the query.

**Definition 8 (Query Variables).** Let \( \mathcal{V} \) be a countably infinite set of query variables ranging over \( \mathbb{UBL} \). A variable \( v \in \mathcal{V} \) is prefixed by a variable identifier ‘?‘ or ‘$‘ (e.g., $?x$).

More formally, the core component of a SPARQL query takes the form of an RDF triple, where the positions \((s, p, o)\) can be either RDF terms or variables. Such triples are called triple patterns, where a user specifies the known RDF terms in the triple pattern, and leaves those that are unknown as wildcards, specifying them as query variables. A set of such triple patterns is referred to as a basic graph pattern (BGP), where the use of the same variables in multiple triple patterns implies a join.

**Definition 9 (Triple Pattern, BGPs).** Given the set of RDF terms denoted by \( \mathcal{T} \), a triple pattern \( tp = (s, p, o) \) is an element of the set \( Q = (\mathcal{T} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{V}) \times (\mathcal{T} \cup \mathcal{V}) \). A basic graph pattern is a set of triple patterns.

The definition of triple patterns indicates that query variables can occur in any position of a triple pattern. The variables of a triple pattern \( tp \) are denoted by \( \text{vars}(tp) \). Following the semantics of SPARQL in [Pérez et al., 2009], we adopt the following definition for valid SPARQL query patterns:

**Definition 10 (A SPARQL query pattern is recursively defined as follows).**

(i) Any BGP is a query pattern, denoted by \( P \).

(ii) If \( P_1 \) and \( P_2 \) are query patterns, then expressions \( (P_1 \text{ AND } P_2) \), \( (P_1 \text{ UNION } P_2) \), and \( (P_1 \text{ OPTIONAL } P_2) \) are also query patterns.

(iii) If \( P \) is a query pattern and \( R \) is a SPARQL built-in condition [Pérez et al., 2009], then \( (P \text{ FILTER } R) \) is also a query pattern.

The operators \text{AND}, \text{UNION} and \text{OPTIONAL} are binary, where \( R \) is a boolean expression that is constructed using elements from the set \( \mathcal{U} \cup \mathcal{L} \cup \mathcal{V} \) and constants, logical connectives \((\neg, \land, \lor)\) and the equality symbol \((=)\) etc., (see [Pérez et al., 2009] for more details). When a SPARQL query is executed over an RDF graph, it will match the known RDF terms with parts of the query RDF graph, where the unknown parts (i.e., the variables) will return a set of bindings, called solutions.
Definition 11 (Solution mapping). Let $\mu$ be a solution mapping from a set of variables to the set of RDF terms, denoted by $\mu : V \rightarrow T$; the domain of $\mu$ is denoted by $\text{dom}(\mu)$. Solution mappings can be applied to triple patterns written as $\mu(tp)$, such that $\mu$ replaces all variables in $tp$ with RDF terms, more formally $\forall x \in \text{dom}(\mu) \cap \text{vars}(tp)$ in $tp$ by $\mu(?x)$.

Each binding pairs a variable with its bound value. The answer of a query that consists of a BGP or a group of BGPs is a join over the results, over individual triple patterns.

Prefix declarations (optional):
BASE <uri>
PREFIX prefix: <uri>

Dataset clause (optional):
FROM <uri> | FROM NAMED <uri>

Result clause (required, select one):
SELECT [DISTINCT|REDUCED] | ASK | CONSTRUCT { BGP } | DESCRIBE

Query clause (optional, required for ASK):
WHERE { query pattern [ FILTER expression ] }

Solution modifier (optional):
ORDER BY
LIMIT n, OFFSET m

Listing 2.4: SPARQL query language structure.

For the purposes of completeness, Listing 2.4 shows the language constructs of SPARQL, along with the different options regarding how a SPARQL query is formulated. Typically, a SPARQL query begins by defining prefixes for URIs, similarly to Turtle syntax. Next, the optional dataset clause specifies the RDF graph or the set of RDF graphs over which the query is to execute. In SPARQL terms, this is known as the SPARQL dataset, which consists of a default graph that does not have a name (i.e., URI), and zero or more named graphs. If the dataset clause is not explicitly stated, then the query is always executed over the default graph.

Definition 12 (SPARQL Named Graphs & Dataset). A named graph is a pair $(u, G)$ where $u$ is a URI that denotes the graph, $u \in \mathcal{U}$, and $G$ is an RDF graph (as by Definition 2). A SPARQL dataset comprises the default graph, which is an RDF graph without a URI, and zero or more named graphs.
The result clause specifies the type of results that should be returned by the specified query. A SPARQL query can return four types of results denoted by the language constructs SELECT [DISTINCT|REDUCED], ASK, CONSTRUCT, and DESCRIBE. Next, the query clause specifies the BGPs or groups of BGPs to be executed, and finally there is a solution modifier section. For a comprehensive discussion on the different features of SPARQL, with a variety of examples for each type of query, the reader is referred to the formalisation of SPARQL in [Harris and Seaborne, 2013].

2.5 Data Integration in Linked Data

Section 2.2 discussed the characteristics introduced of the LD paradigm, such as the use of RDF as a uniform model for describing the data, de-referencability that allows entities to be discovered on the Web along with semantic knowledge that describe resources using semantic annotations in formal SW languages, as well as typed RDF links that allow data to be interlinked. However, despite these features, several challenges exist in terms of consuming the WoD. An effective exploitation of the WoD for consumers poses several challenges that fall into the following non-exhaustive categories. We note that the contributions described in this dissertation are mostly concerned with heterogeneity.

Usability: The WoD follows the Web of documents in providing a simple search-browse model for discovering information by following links. It is hard, however, for the end user to locate the RDF graphs that can potentially contribute answers to their query requests. Semantic web search engines such as Swoogle [Ding et al., 2004] and Watson [d’Aquín et al., 2007] crawl and index the WoD, providing keyword search capabilities. However, the WoD should have the potential for more precise query answering by using structured queries expressed in SPARQL.

Heterogeneity: The WoD could be construed as a large-scale distributed database that can be semantically integrated at both the instance and conceptual levels, thereby providing a unified integrated view over a diverse collection of WoD sources [Hausenblas and Karnstedt, 2010; Heath and Bizer, 2011]. However, the low publication barriers, as well as the diversity of publishers without central coordination, causes several discrepancies to exist at
both the instance and conceptual levels. For example, at the instance level, one may experience different Web resources being used to represent the same real world-entities (or abstract concepts), redundant information, contradictions in predicate values and format inconsistencies in value representations. At the conceptual level, one may experience different conceptualisations of the same domain, inconsistencies in the structural representation of concepts and in terminologies from LD vocabularies, etc. Recently, there has been an effort to alleviate some of these heterogeneity issues with the use of links published as additional RDF statements, e.g., owl:sameAs links at the instance level and owl:equivalentClass / owl:equivalentProperty or rdfs:subClassOf / rdfs:subPropertyOf for the conceptual level. Such integration efforts require the coordination of publishers or communities, and therefore could be considered a laborious, partial solution.

This chapter argues that techniques developed for pay-as-you-go data integration in databases could be used to mitigate heterogeneity conflicts in WoD sources. Pay-as-you-go data integration, would allow incremental integration of RDF data sources building on automated reconciliation of the different kinds of semantic heterogeneities.

**Scalability:** The amount of structured data on the Web is increasing, with increasing numbers of publishers adopting the LD paradigm to make their datasets publicly available. Clearly, approaches for consuming the WoD should take this scale and continuous change into account.

Several proposals that relate to the integration of data from multiple RDF data sources are under ongoing investigation that take into consideration some or all of the above challenges. The remainder of this section presents a review of representative research proposals related to combining data from multiple RDF data sources, offering distributed query processing capabilities, and data integration tasks such as semantic matching or mapping on the WoD. In addition, where possible, we discuss how different proposals could potentially benefit from pay-as-you-go data integration techniques or principles. Note that this section does not describe research that might be considered to compete with the contributions presented in this research: this is done inside the contribution chapters.
2.5.1 Federated Query Processing

As previously discussed, the LD paradigm builds on several standards that enable data to be published in machine-readable form. The consistent use of the RDF model for describing the data and the use of SPARQL provide the opportunity for the WoD to act as a large decentralised and distributed database. A detailed review of research proposals and concepts for querying RDF data sources was published in [Hose et al., 2011]; the reader is referred to that study for a complete account of the different proposals. This section is concerned with the most prevalent federated/distributed query processing proposals.

Taking into account the increasing availability of LD datasets exposed as SPARQL endpoints\(^\text{15}\) (with varying degree of overlap), research on federated SPARQL querying is gaining attention from the relevant community. Many research proposals, including, SemWIQ [Langegger et al., 2008], DARQ [Quilitz and Leser, 2008], SPLENDID [Görlitz and Staab, 2011] and FedX [Schwarte et al., 2011a] are based on a mediator-based approach to execute queries over a federation of endpoints. Typically, this process is concerned with deciding which parts of the query are best routed to which endpoint, creating multiple sub-queries (typically, for each basic graph pattern), sending the relevant part to the appropriate endpoints that participate in the federation, and subsequently combining the results into the final query result.

The primary challenge for such SPARQL query engines is having a-priori knowledge as to which participating endpoints can contribute which results. A common practice for such systems is for them to rely on the existence of service descriptions or catalogues that provide description of the contents that can be found in each endpoint, as well as various statistics that are then used during query optimisation. For example, DARQ provides transparent access to distributed endpoints relying on the existence of service descriptions for each of the SPARQL endpoints. In a complementary proposal, SPLENDID uses statistics from voID\(^\text{16}\) descriptions as input to a query optimisation strategy for federating SPARQL endpoints.

Alternative approaches exist. For example, instead of relying (entirely) on the existence of service descriptions that are pre-defined, FedX [Schwarte et al.,

\(^1\text{http://stats.lod2.eu/}\)

\(^1\text{The Vocabulary of Interlinked Datasets (voID) is an RDF Schema vocabulary used to provide meta data descriptions of LD datasets.}\)
2011a] and SPLENDID [Görlitz and Staab, 2011] propose the use of SPARQL ASK queries for source selection. The extension of SPARQL 1.1 with federation features motivated the proposal of federated query engines that make use of such features [Aranda et al., 2011].

2.5.2 WoD as a Global Distributed Dataspace

Research proposals in this category view the WoD from the perspective of a dataspace [Heath and Bizer, 2011] comprising of distributed RDF graphs where links play a key role, both for querying and for easing the various kinds of semantic heterogeneities that exist on the WoD. With respect to querying, this perspective on the WoD suggests that queries could be answered directly on the WoD with the use of links. Briefly, the basic idea behind approaches that seek to query the WoD directly is to discover sources that might contain query-relevant data, and during query execution time to iteratively fetch data from the discovered sources using HTTP URI lookups.

In contrast to the federated approaches over SPARQL endpoints, such techniques do not require sources to provide a query service (i.e., a SPARQL endpoint). One such approach was proposed by Hartig et al. [2009]. It relies on links for querying over raw data sources. To evaluate a SPARQL query over the WoD, the system dereferences URIs from the graph patterns specified in the query to create an initial dataset that can be used to partially evaluate the query. It is probably the case that, during this process, further links will be discovered that are traversed in an iterative fashion to derive a more complete dataset that will eventually satisfy the query. The search process of query-relevant data from URI lookups is recursive, and it terminates when there are no more potential links to follow.

Such approaches for evaluating queries directly on the WoD are dependent on the existence of RDF links in the RDF graphs discovered, and are often vulnerable to limitations, such as retrieving unforeseen large RDF graphs, infinite link discovery, slow response times, and poor results due to unhelpful or irrelevant initial links [Hartig, 2011]. The link traversal query execution method introduced by Hartig et al. [2009] seems not to make use of identity links expressed as typed owl:sameAs links, or vocabulary links that map across vocabularies (e.g., owl:equivalentClass) for retrieving semantically-related results.

A complementary approach is the proposal by Umbrich et al. [2014]. It takes
into account some of the semantics of RDFS and OWL to infer additional query-relevant sources. Pay-as-you-go data integration techniques can be adopted for the automatic discovery of semantic correspondences (i.e., matches) between pairs of LD datasets that can give rise to RDF links. For example, techniques from schema matching [Rahm and Bernstein, 2001], and record linkage [Elmagarmid et al., 2007] can be used to discover identity links or vocabulary mappings (see Section 2.2.4) on the WoD. At the instance level, such techniques can be used to discover \texttt{owl:sameAs} links; a task that, in the WoD context, has been called \textit{co-reference resolution} [Ferrara et al., 2011]. Moreover, at the conceptual level, such techniques can be adopted for discovering semantic correspondences between LD vocabularies (or ontologies). Here, we briefly present representative proposals inspired by data integration research, beginning with those that discover \texttt{owl:sameAs} links at the instance level.

\textit{Silk} [Isele et al., 2010a]: This is an approach for interlinking LD datasets and maintaining such links. It consists of an identity resolution component to augment the WoD with additional RDF links that refer to the same real-world objects, and a specification language that is used to specify the conditions which determine whether an entity is known to the system. The identity resolution component takes as input a pair of SPARQL endpoints that are specified in a file, along with various parameterised matching techniques (e.g., string matching, numeric equality, taxonomical distance similarity) to derive a similarity score that estimates the equivalence of the matched entities. In addition, the user can specify data transformation functions (called linkage rules), e.g., concatenating the values of predicates, that are used to improve the matching. An improved version of Silk, which combines genetic programming and active learning in order to learn expressive linkage rules, is described in [Isele and Bizer, 2013]. Moreover, several techniques are used to combine the confidence scores from each of the matching techniques used, such as \texttt{MIN}, \texttt{MAX} and \texttt{AVG}. Silk is able to hypothesise \texttt{owl:sameAs} RDF triples.

\textit{ObjectCoref} [Hu et al., 2011]: This is a tool that allows the discovery of identity links between Web resources that are published on the WoD. This approach proposes a semi-supervised learning framework. It requires an initial training set of reference links that are then used to learn discriminative
property-value pairs. ObjectCoref, therefore, needs training before being able to discover any identity links. In so doing, it also exploits semantic knowledge such as existing \texttt{owl:sameAs} links, (inverse) functional properties and cardinality restrictions to compute a discriminating factor for pairs of properties between the matching resources. Thereafter, identity links are derived based on the matching properties following an iterative process.

This dissertation is not concerned with the challenge of identifying identity links, published as \texttt{owl:sameAs} on the WoD. Therefore, for a recent survey, the interested reader is referred to [Ferrara et al., 2011]. Instead, our contributions assume that, as the WoD establishes itself, more identity links (as well as other types of semantic annotations) will be more prevalent.

To enable applications to integrate data by following links on the WoD, heterogeneities at both the instance and the conceptual levels need to be discovered and resolved. On the WoD, the general practice for doing this is to publish typed links that are annotated with formal semantics (e.g., transitivity, symmetry etc.) so that more knowledge can be inferred using reasoners [Polleres et al., 2013].

We now shift our focus to proposals that discover vocabulary links at the conceptual level.

\textit{BLOOMS} [Jain et al., 2010a]: This proposal deals with the problem of identifying vocabulary links (also referred to as \textit{alignments}) between ontologies of LD datasets. BLOOMS is able to discover both \texttt{owl:equivalentClass} and \texttt{rdfs:subClassOf} links between classes of LD ontologies. For the purpose of finding such alignments, BLOOMS uses a central forest of concepts derived from topics in Wikipedia.

\textit{Parundekar et al. [2013]}: This proposal describe an approach for finding alignments between classes from multiple LD datasets with the goal of increasing the interoperability of ontologies from the WoD that were previously disconnected. To generate alignments between a pair of classes from two ontologies, the degree of overlap between the extensions of complex or atomic classes is measured. Based on the measurement, the type of the alignment is decided: whether the pair of classes are related by subsumption, or equivalence.

The publication of RDF links on the WoD shares commonalities with semantic mappings in the sense that both could be used to reconcile the various forms of
heterogeneities are likely to arise in the sources. Specifically, mapping techniques can be used to provide more expressive information regarding how data from potentially similar resources can be translated into a form that an LD application can understand. For example, mappings can deal with specifics such as structural transformations and property value transformations. Moreover, due to the fact that the WoD is an open environment with a diversity of publishers, there is an inherent uncertainty associated with identity links or vocabulary mappings often discovered with automatic techniques, such as those mentioned above. Such links are dependent on the semantics of a particular application domain, and still require manual verification of the links discovered.

With respect to pay-as-you-go data integration principles, feedback can potentially be useful for informing revisions of automatically generated identity links or vocabulary mappings. Publishers that have knowledge of a specific domain can be considered as candidates for providing such feedback. There is recent work on discovering and publishing expressive mappings on the WoD [Bizer and Schultz, 2010], and on using user feedback [ul Hassan et al., 2012] for verifying discovered identity links within applications that are consuming the WoD. However, the opportunities that are available by using feedback to guide the revisions of such RDF links are far from being fully explored.

2.5.3 Aggregation of Search Results

Semantic Web search engines, such as Swoogle [Ding et al., 2004] and Watson [d’Aquin et al., 2007], crawl and index the WoD providing keyword search capabilities. For a given keyword query, the goal is to return a list of ranked resources matching the keyword phrase with the descriptions of entities, published as Web resources, to determine their relevance. In contrast, approaches such as Sig.ma [Tummarello et al., 2010] use such search results (in this particular case, those obtained through Sindice [Oren et al., 2008]) to collect structured data in RDF, which is then aggregated to form an estimated entity profile. Sig.ma provides a holistic approach for data discovery and consolidation, which brings together data from multiple resource descriptions to create entity profiles. If descriptions of resources use the same predicates, then their values are merged together. If there is an owl:sameAs predicate, then the object URI is followed to retrieve more information about the entity to be described. In addition, users are allowed to interact with the generated page either by navigating to other
pieces of information using the discovered URIs, or by refining the list of discovered sources that participate in the results. The refinement capabilities offered by Sig.ma, e.g., by accepting or rejecting specific sources that contribute information to the aggregated pages, can be considered as a form of pay-as-you-go integration.

However, Sig.ma does not resolve any of the semantic heterogeneities that the data sources participating in the mash-up might exhibit. Sig.ma can be characterised as an aggregated-search approach, insofar as it does not offer a semantically-consistent view over multiple data sources. Potentially, Sig.ma could offer users the opportunity to create custom-built tables (i.e., views) of information by specifying the desirable predicates, somewhat akin to a mediated schema tailored for entity profiles. Then, techniques from pay-as-you-go data integration, such as matching and mapping, could be developed that allow such tables to be populated with data. For example, SPARQL \texttt{CONSTRUCT} queries could be used to translate data from the sources discovered and to publish them to the required format specified by the table. In such a scenario, and because pay-as-you-go techniques build on automation, it is likely that the desired results might not meet the requirements of the user. In such cases, Sig.ma could be extended to collect feedback to refine the matchings or the mappings derived, to revise the results. However, currently, Sig.ma does not use these more sophisticated form of integration techniques.

## 2.6 Dataspaces over Linked Data

As previously discussed, the objective of data integration is to offer a semantically-consistent view over multiple data sources that participate in an integration scenario. Typically, a data integration system involves the creation of an integration schema\footnote{Integration schema is often referred to as mediated or global schema.} which is exposed to the user (or application). The purpose of the integration schema is to provide a unified representation against which queries can be posed, relieving users from the need to know the structure of the underlying sources, or the need to query them manually and combine the multiple results.

Assuming the relational model for a brief moment, the integration schema does not need to include all the concepts, attributes and relationships expressed in the data sources, nor does it need to include their intersection [Sarma et al.,
Rather, it can capture only the terms that fulfill the requirements of the integration designer. Although there may not be a priori single integration schema to which data from the sources conform, the system simply needs to know the exact relationships between the schemas of the sources and the integrated schema. This suffices for a traditional data integration system to be able to provide services to the user [Franklin et al., 2005]. These relationships are represented in terms of mappings [Lenzerini, 2002]. In LD, semantic mappings can be specified in terms of SPARQL queries. A full account of how to specify mappings that translate data from RDF data sources into a form that is compatible with the integration schema in the context of integrating LD is beyond the scope of this dissertation and the interested reader is referred to research proposals on using SPARQL CONSTRUCT queries [Etcheverry and Vaisman, 2012] or SPARQL query rewriting [Correndo et al., 2010].

Typically, a traditional data integration system involves the following processes:

- **matching**, which enables the discovery of semantic correspondences at both schema and instance levels by quantifying the similarities between elements of the schema or instances (e.g., through a similarity score). In the literature, there are numerous research proposals for semi-automatically discovering such semantic correspondences in the context of schema matching [Belahsene et al., 2011; Bernstein et al., 2011], as well as that of ontology alignment [Euzenat et al., 2007; Shvaiko and Euzenat, 2013].

- **mapping**, which involves providing the system with expressiveness (e.g., views) that can be used to reformulate the queries posed against the mediated schema into a set of queries over the data sources. In a traditional data integration system, the creation of mappings involves human input to resolve the imperfections and shortcomings of the semantic correspondences discovered, as well as to refine the mappings if the process is supported by a tool (e.g., [McCann et al., 2005] or [Bonifati et al., 2008]).

Traditional data integration approaches are likely to fall short in terms of cost-effectiveness due to the laborious and error-prone process required to set up and maintain the system. This means that such approaches are best suited to the integration of data sources that are of a manageable size under stable environments. Hedeler et al. [2009] point out that traditional data integration
offers high-quality, high-cost integration, which makes this process less effective for a large number of sources that are potentially changing over time, or for on-demand integrations. In dynamic environments such as the WoD, where it is likely that RDF data sources are changing over time, the relevant data is likely to be distributed across many RDF data sources (potentially in different domains). Also, user requirements are less predictable. In the WoD, traditional data integration seems not as appropriate as one might have wished [Paton et al., 2012].

Section 1.3 made the point that, in contrast to traditional data integration systems, the vision of dataspaces suggests that the resource-intensive process of producing an integration of sources can be replaced at a much lower cost if the system simultaneously offers opportunities for incremental refinement and improvement using a pay-as-you-go approach to integration. According to [Hedeler et al., 2010b], a dataspace typically has a life-cycle involving the following phases:

(i) *bootstrapping*, i.e., the automatic initialisation of the system, where the system postulates semantic correspondences between the sources and uses them to derive mappings between the schemas of the participating data sources and the integration schema. The integration schema is either provided to the system, or it can be automatically derived from the schemas of the sources (e.g., by union [Salles et al., 2007] or by merging them). Thus, the mediated schema can contain concepts that appear in numerous sources [Sarma et al., 2008]. Where the aim of the bootstrapping phase is to provide a unified point of access for the users to pose queries (either structured or keyword-based), mappings that are derived automatically are likely to be imprecise because they build upon uncertain matches.

(ii) *usage*, i.e., evaluation of user requests, where best-effort query services are provided as early as possible, using the mappings derived during the bootstrapping. The initial integration is a speculative integration and, because of this, it potentially has low quality, leading the system to produce less accurate results than would be the case if human experts were involved. However, the early usage of even a speculative integration offers potential for improvement so as to evolve the latter; and

(iii) *improvement*, i.e., the collection of feedback in the light of *usage*, which is the payment (so to speak) for using the system. Feedback is used to refine
the quality of the integration e.g., filtering out poor semantic correspondences, thus improving the quality of the mappings derived from them. In this phase, the system therefore responds to user feedback, which can take several forms (e.g., on the results of queries over the integrated schema [Belhajjame et al., 2013]). Over time, the idea is that the system swiftly and cost-effectively improves the overall integration, thereby compensating for the extensive usage of automation during the bootstrapping phase.

To support the ethos of a bootstrapped system that builds on automation, dataspaces should be able to handle of uncertainty throughout the life-cycle [Sarma et al., 2011; Kuicheu et al., 2013]. Section 2.6.1 elaborates on how the life-cycle of a dataspace system might work over LD sources and introduces a case study to illustrate how pay-as-you-go techniques can be adopted for the integration of multiple, possibly heterogeneous, RDF data sources. Note that although, not all challenges have been resolved and reported as contributions to this dissertation, the study elaborates on all the life-cycle phases which motivated the reported research contributions. Section 2.6.2 reviews recent proposals on managing uncertainty in pay-as-you-go data integration systems.

2.6.1 Life-cycle of a Dataspace over LD

In the literature there are several proposals that suggest different methodologies for pay-as-you-go integration. A relatively recent survey by [Hedeler et al., 2010b] identified that proposals differ in their approach to the life-cycle phases. This section presents a case study to elaborate on the impact and challenges of adopting the life-cycle phases of a dataspace to support querying over multiple RDF data sources. As part of our discussion, the study follows the architecture of a real dataspace management system, specifically the Manchester DSToolkit [Hedeler et al., 2012]. A pair of real RDF data sources, from the same music domain, are used in this study. Specifically, the RDF data sources are available under the DBTune.org project, viz., Jamendo\(^{18}\) and Mangatune\(^{19}\).

To provide transparent access to multiple data sources, data integration is concerned with reconciling the various heterogeneities that the sources might exhibit, at both the instance and the conceptual levels. Research proposals for

\(^{18}\)http://dbtune.org/jamendo; retr. 2014/01/04

\(^{19}\)http://dbtune.org/mangatune; retr. 2014/01/04
solving this challenges at the conceptual level, utilise conceptual descriptions (i.e., schemas) of the data sources to discover and reconcile the various structural and semantic heterogeneities of the data sources at the conceptual level. In the context of LD, the requirement for a schema is more relaxed, since there is a lack of a conceptual description of datasets to which all the data conform [Jain et al., 2010b]. For example, there is no mechanism to describe that Jamendo captures concepts related to music or on how such concepts are structured.

As discussed in Section 2.5.1, approaches to federated query processing have made steps in this direction by providing source descriptions or void descriptions [Alexander et al., 2009]. However, such approaches focus more on describing statistics about an RDF data source and do not provide complete descriptions of how concepts are structured. In data integration, the knowledge of schemas is important for discovering semantic correspondences that give rise to mappings that, in turn, capture the relationships between the schemas of the sources and the integrated schema. As we shall see in the case study outlined in this section, it seems that, for the integration of RDF data sources, conceptual descriptions are useful for: (i) the derivation of mappings expressed in a way that enables the translation of data from the sources, so that they can be used to populate the integration schema; and (ii) for providing a description of the structure of the RDF graphs, so that complex graph patterns can be formulated over the RDF data source being queried using SPARQL.

**DSToolkit: A Dataspace Management System**

*DSToolkit* is a library of data integration techniques and a dataspace management system (DSMS) that supports the life-cycle phases. In particular, DSToolkit provides users with techniques to: (i) select data sources they wish to integrate, (ii) discover semantic matches, (iii) derive semantic mappings from the matches discovered [Mao et al., 2009], (iv) querying capabilities using SMql a declarative query language inspired by SQL [Hedeler and Paton, 2011], and (v) select and refine the discovered mappings by utilising user feedback, used to annotate query results [Belhajjame et al., 2013]. In general, DSToolkit supports different styles of integration, either utilising an integration schema that is manually provided to the system, or by inferring an integration schema automatically by operating on available schemas. The following section describes a DSToolkit case study for the integration of RDF data sources when an integration schema is provided.
As regards implementation details, DSToolkit builds on model management [Bernstein and Melnik, 2007], thus enabling data integration techniques to be developed and used in a generic way that is agnostic to the different data models (e.g., relational, object-oriented, XML) of the data sources to be integrated. Table 2.2 describes the model-generic constructs\(^{20}\) comprising the MISM [Atzeni et al., 2009a,b] metamodel, and the more generalised constructs that implement DSToolkit’s canonical model, which further generalises the MISM constructs. In addition, the table describes some of the model-specific constructs, and their mapping to the canonical model of DSToolkit. For a complete list of the models supported, the reader is referred to [Hedeler et al., 2012]. To extend DSToolkit so that it enables the integration of RDF data sources, an importer was implemented that supports a subset of the SWIM metamodel constructs proposed by Virgilio et al. [2009]. The SWIM metamodel defines CLASS to represent an RDFS or OWL concept of a class, PROPERTY to represent the concept of literal RDF triples, PREDICATE to represent the concept of RDF links and CONTAINER to represent the concept of blank nodes.

![Table 2.2: Summary of model-generic and model-specific constructs.](attachment:image.png)

\(^{20}\)The term construct here is used to refer to the building blocks (i.e., concepts) that comprise each model.
Case Study: Integrate RDF Data Sources

The case study described in this section assumes the existence of an LD application for integrating the Jamendo and Magnatune RDF graphs, thus providing seamless querying capabilities. To fulfil this requirement, the application developer manually designs an integration schema that collects information about musicians, and their released records, each consisting of a list of tracks and of pricing information for each track. In more detail, Figure 2.2 illustrates the integration schema represented as an entity-relationship (ER) diagram; where RDF classes are expressed as entity-types, literal RDF triples represent attributes, and RDF links express binary-relationships. Note that our hypothetical integration schema comprises RDF terms collected from the Music Ontology (MO), FOAF, the Dublin Core (DC)\(^{21}\) vocabularies as well as terms from a proprietary vocabulary that uses the \textit{ns-x} namespace. On the WoD, publishers have the freedom to mix terms from various LD vocabularies, as well as the option of creating their own RDF terms, to describe their data. In Figure 2.2, the application developer defines \textit{ns-x:date\_created} and \textit{ns-x:track\_title} as sub-properties of \textit{dc:date} and \textit{dc:title} respectively (shown as grey circles).

![Integration schema](image)

Figure 2.2: Integration schema used for the integration of RDF data sources from DBTune.org.

\(^{21}\)http://purl.org/dc/elements/1.1/
For the case study we assume the existence of conceptual descriptions for each of the participating LD sources. Figure 2.3 shows examples of such conceptual descriptions, that can be inferred from RDF data sources so as to support tasks such as matching, mapping and query rewriting, as described later in this section. From the conceptual descriptions, one can observe that, despite the fact that Jamendo and Magnatune are published under the DBTune.org project using the music ontology, the datasets do not strictly adhere to it. Each source has its own peculiarities in terms of structure and semantic information captured, thus exhibiting various forms of heterogeneity. One such example, in Figure 2.3(a), is the binary relationships foaf:maker, and its inverse relationship, foaf:made. Such relationships are used in the Jamendo dataset to capture authorship information about authors (i.e., mo:MusicArtist) of a record (i.e., mo:Records). Contrastingly, such information is captured using only the foaf:maker relationship in the Magnatune dataset (see Figure 2.3(b)).

Listing 2.5: SPARQL query that retrieves authorship information about Artists and their Records.
We now illustrate, using a concrete example, how authorship information about artists and records can be captured in an integration schema that uses only the `foaf:maker` predicate. The heterogeneity between the data sources can be resolved with a mapping expressed using SPARQL (see Listing 2.5).

To write a SPARQL query that provides a uniform representation over the different realisations of the authorship relation between artists and records, it is essential for the user to: (i) understand the application domain; and (ii) have precise knowledge of the conceptual description of each data source.

**Bootstrapping Phase:**

Integrate RDF data sources $d_1$ and $d_2$ by matching them with the integration schema $s_{int}$, infer schematic correspondences and generate the corresponding mappings.

1: $s_1 = \text{INFERSHEMA}(d_1)$
2: $s_2 = \text{INFERSHEMA}(d_2)$
3: $MT_{s_1-s_{int}} = \text{MATCH}(s_1, s_{int})$
4: $MT_{s_2-s_{int}} = \text{MATCH}(s_2, s_{int})$
5: $CR_{s_1-s_{int}} = \text{INFERCORRESPONDENCE}(MT_{s_1-s_{int}})$
6: $CR_{s_2-s_{int}} = \text{INFERCORRESPONDENCE}(MT_{s_2-s_{int}})$
7: $MP_{s_s-s_{int}} = \text{VIEWGEN}(s_{int}, \{s_1, s_2\}, \{CR_{s_1-s_{int}}, CR_{s_2-s_{int}}\})$

**Usage Phase:**

Querying by formulating queries using over the integration query.

8: $R_{qs_{int}} = \text{ANSWERQUERY}(qs_{int}, MP_{s_s-s_{int}})$

9: **LOOP**

**Improvement Phase:**

User provides feedback by annotating query results using some annotation form, denoted by $A$, which are then used to annotate the mappings with estimates of precision and recall.

10: $R_{qs_{int}}^{an} = \text{ANNOTATE}(R_{qs_{int}}, A)$
11: $MP_{s_s-s_{int}}^{an} = \text{ANNOTATE}(MP_{s_s-s_{int}}, R_{qs_{int}}^{an})$

Usage: by using annotated mappings to produce query results.

12: $CP = (\text{threshold}, \text{precisionTarget})$
13: $R_{qs_{int}} = \text{ANSWERQUERY}(qs_{int}, CP, MP_{s_s-s_{int}}^{an})$

14: **END LOOP**

Listing 2.6: Case scenario integration steps using DSToolkit.

Complex SPARQL queries, such as in Listing 2.5, make the process of reconciling heterogeneities from multiple RDF data sources, complex and error-prone. A
dataspace system builds on automation to resolve the various kinds of heterogeneity by deriving mappings from postulated construct equivalence, i.e., matches. For the purposes of the case study, we have adopted the model management operators in [Hedeler et al., 2012] with some modifications and additions so that they can work over RDF graphs. For the specificities and semantics of the DSToolkit’s model management operators, the interested reader is referred to [Hedeler et al., 2010a] although this is not essential for understanding the contributions described in this dissertation. When necessary and for the purposes of discussing the experiment, we explicitly refer to them below.

Listing 2.6 shows the steps required for the integration and querying of RDF data sources (under this scenario) using an instantiation of the DSToolkit framework. For this case study, we assume that the integration schema $s_{int}$ is provided, and that the RDF data sources Jamendo, denoted by $d_1$, with conceptual description $s_1$, and Magnatune, denoted by $d_2$, with conceptual description $s_2$, are to be integrated. We now explain the steps that enable the automatic initialisation and querying of a dataspace over heterogeneous RDF data sources.

**Conceptual Descriptions**

In the context of LD, there is a lack of conceptual descriptions of LD datasets. Although terminology from LD vocabularies is used to annotate RDF graphs, an RDF data source potentially contains terms from several such vocabularies structured in different ways. In Section 2.5.1, we reviewed approaches for distributed query processing, such as DARQ, that required knowledge of the RDF predicates to be included in the sources in order to identify which sources can contribute answers to which triple patterns. This indicates that there is a need for research into techniques that infer conceptual descriptions over RDF data sources.

In RDF data sources, the notion of a conceptual description similarly to a schema from relational databases is a more relaxed one. As previously discussed, data integration systems tend to require that conceptual descriptions be available. Typically, information from conceptual descriptions is used for discovering semantic correspondences, and for the derivation of mappings that reconcile the heterogeneities of the sources, insofar as they are executable expressions that translate source data into integrated data.

In the case study, conceptual descriptions needed to be inferred for each of the RDF data sources. This is done in lines 1-2 of Listing 2.6, where the integration
process starts by invoking the INFERSHEMA operator. The outcome of this operation is a set of schema constructs, denoted by $C_s$, that belong to schema $s$. More formally, the signature of the INFERSHEMA operator is, $\text{INFERSHEMA} : d_i \rightarrow C_s$ (following the formalisms described in [Hedeler et al., 2010a]), where $C_s$ is a set of constructs and $d_i$ a set of instance data.

If they are not already available, it is crucial for the bootstrapping phase of a system that supports a pay-as-you-go type of integration that such conceptual descriptions can be inferred automatically. Figure 2.3 shows examples of such conceptual descriptions that have been inferred automatically from the participating RDF data sources, using the contributed technique for schema inference described in Chapter 3.

**Identification of Matches**

Semantic correspondences as to how the integration schema relates to the source schemas need to be made explicit before the mappings that translate data from the sources into a format that populates the integration schema can be derived. Typically, it is essential for a data integration system to use information at both the schema and the instance levels. Here, we focus only on the discovery of correspondences using schema level information. Figure 2.4 shows a subset of semantic correspondences discovered using string-based matching techniques (e.g., n-gram) over the Jamendo and Magnatune data. In the literature, there is a considerable body of work, especially from schema matching in databases and ontology alignment, for discovering such correspondences at different levels using different sources of evidence that might help quantify any similarities. Often, such semantic correspondences are construed as associations between the schema elements along with a numeric score (e.g., a similarity score, or a probability [Nottelmann and Straccia, 2005]) to characterise, as it were, the degree of confidence on the equivalence between a pair of schema elements.

\[
\begin{align*}
mt_1 : & \langle s_1.\text{Record}, s_{\text{int}}.\text{Record}, 1.0 \rangle \\
mt_2 : & \langle s_1.\text{Record.title}, s_{\text{int}}.\text{Record.tracktitle}, 0.54 \rangle \\
mt_3 : & \langle s_1.\text{Tag.tagName}, s_{\text{int}}.\text{MusicArtist.name}, 0.45 \rangle \\
mt_4 : & \langle s_2.\text{Performance.recorded.as}, s_{\text{int}}.\text{Record}, 0.6 \rangle
\end{align*}
\]

Listing 2.7: Example of schema matching results.
Figure 2.3: Inferred conceptual descriptions of DBTune.org RDF data sources represented as entity-relationship diagrams.
The integration process proceeds with the MATCH operation (lines 3-4 of Listing 2.6). The inferred conceptual descriptions, $s_1$ for Jamendo and $s_2$ for Magnatune, are matched with the integration schema $s_{int}$ to derive the set $MT_{s_i,s_{int}}$ of matches. In DSToolkit, a match $mt_{s_i,s_j} = \langle C_{s_i}, C_{s_j}, conf\,Score \rangle$ is an association between constructs of the schemas.

We note the reader that the contributions discussed in this dissertation focus on a technique that infers a conceptual structure by observing an RDF source (the INFERSCHEMA operator as described in lines 1-2 of Listing 2.6) and on techniques that take advantage of the rich semantic annotations as sources of evidence to revise the decisions of matching techniques that typically build on string-based comparisons alone (the MATCH operator as described in lines 3-4 of Listing 2.6).

**Expressive Schematic Correspondences**

As is commonly the case with matching techniques, numerous matches are discovered between the schemas that require filtering and grouping in order to inform the derivation of the semantic mappings. However, matches frequently lack sufficient information for the automatic derivation of complex mappings that describe how data are to be transformed. To bridge this gap, Guo et al. [2013] proposed a technique to discover richer match definitions, known as schematic correspondences, based on the classification of correspondences identified by Kim and Seo [1991] – examples include 1-to-1, 1-to-many and many-to-many correspondences.

In DSToolkit, the operator INFERCORRESPONDENCE (lines 5-6 of Listing 2.6) infers the set of schematic correspondences, denoted by $CR_{s_i,s_{int}}$, between each of the source schemas and the integration schema. A schematic correspondence is expressed for each pair of constructs from the schemas annotated with the kind of relationship identified – e.g., ‘missing attribute’, ‘horizontal’ or ‘vertical’ partitioning. Following the case study, Figure 2.5 shows the most highly-ranked collection of schematic correspondences when applied over the integration schema and the schemas of the local RDF data sources. For example, the semantic correspondences discovered suggest that vertical partitioning exists between $s_{int}.Record$ in the integration schema and $s_1.Record$ and $s_1.Track$. This is indeed true, since $s_{int}.Record$ consists of information that is split across the $s_1.Record$ and $s_1.Track$ classes. One can reconstruct $s_{int}.Record$ by joining the two classes.

This dissertation is not concerned with contributing a proposal on richer definitions of matches, such as schematic correspondences. We discuss this here for
Figure 2.4: Schema matching results.

Figure 2.5: Schematic correspondences inference.
the purposes of completeness, and as part of the case study that shows how a DSMS such as DSToolkit can be adapted towards the integration of RDF data sources from the WoD. However, we note that there might exist the potential for discovering more expressive vocabulary or identity links on the WoD using richer match definitions. For example, publishing information on the Web that indicates \( \text{ns-x:track} \text{title} \) is the same property as \( \text{dc:title} \) with a different name, thus enabling renaming in SPARQL queries while querying over a corresponding RDF graph.

**Derivation of Mappings**

The derivation of the schematic correspondences enables mappings to be generated automatically. There are many approaches in the literature to discover mappings (such as in Figure 2.4) directly from matches (e.g., [Bonifati et al., 2008]). However, since we are adopting the DSToolkit approach in this case study, mappings are inferred from the schematic correspondences derived [Mao et al., 2009]. In DSToolkit, *mappings* are expressed using SMql; a query language that was designed to work over DSToolkit’s canonical model (see Table 2.2). Listing 2.8 shows examples of candidate semantic mappings expressed using SMql. According to [Hedeler et al., 2010a], a mapping \( mp_{s_s \rightarrow s_{int}} = \langle c_{s_{int}}, q_{s_s} \rangle \) is an executable expression (in this case, in SMql) that relates a concept \( c_{s_{int}} \) in the integration schema to a query \( q_{s_s} \) over the set of source schemas \( S_s \) whose results allow the concepts in the integration schema to be populated with instance data from \( S_s \). A set of candidate mappings is denoted as \( MP_{s_s \rightarrow s_{int}} \) and can be generated automatically by invoking the VIEWGEN operator (line 7 of Listing 2.6).

Queries expressed over the canonical model of DSToolkit using SMql can be translated algorithmically to source-specific query languages in underlying RDF data sources, following a methodology similar to that described in [Hedeler and Paton, 2011]. Listing 2.9 shows examples of the translation from SMql queries to equivalent SPARQL ones. Dealing with the semantics of the query languages and how the translation can be performed are beyond the scope of this dissertation.

As discussed in [Heath and Bizer, 2011], the integration aspect on the WoD is usually addressed by an approach based on the links contained in RDF documents. It is worth mentioning that there seems to be an opportunity for utilising such links for the derivation of executable mappings, e.g., in SPARQL. Although
CHAPTER 2. ANATOMY OF DATASPACES OVER LINKED DATA

\[ m_1 : (s_{int}.Record, SELECT R.title as title, R.maker as maker, NULL as description, NULL as date_created, T.title as track_title, T.paid_download as paid_download FROM s_2.Record R, s_2.Track T WHERE R.track = T.title) \]

\[ m_2 : (s_{int}.Record, SELECT R.title as title, R.maker as maker, R.description as description, R.date as date_created, T.title as track_title, T.paid_download as paid_download FROM s_1.Record R, s_1.Track T WHERE R.track = T.title) \]

\[ m_3 : (s_{int}.MusicArtist, SELECT M.name as name, M.img as img, NULL as biography, M.homepage as homepage, M.basednear as based_near FROM s_2.MusicArtist M) \]

\[ m_4 : (s_{int}.MusicArtist, SELECT M.name as name, M.img as img, M.biography as biography, M.homepage as homepage, M.basednear as based_near FROM s_1.MusicArtist M) \]

\[ m_5 : (s_{int}.Record, SELECT T.title as title, T.maker as maker, NULL as description, T.created as date_created, T.title as track_title, T.paid_download as paid_download FROM s_2.Track T) \]

\[ m_n : (...) \]

Listing 2.8: Examples of mappings between \( s_{int} \) and \( s_1, s_2 \).

\[ \]

\[ m_1 : \]

\[ \text{SELECT} \ ?\text{title} \ ?\text{maker} \ ?\text{track_title} \ ?\text{paid_download} \]

\[ \text{WHERE} \{ \]

\[ ?s2 \text{rdf:type mo:Record; } \]

\[ \text{dc:title} \ ?\text{title}; \]

\[ \text{foaf:maker} \ ?\text{maker}; \]

\[ \text{mo:track} \ ?t . \]

\[ ?t \text{rdf:type mo:Track;} \]

\[ \text{dc:title} \ ?\text{track_title}; \]

\[ \text{mo:paid_download} \ ?\text{paid_download}. \]

\[ \}

Listing 2.9: Example of SPARQL-generated query translated from \( m_1 \) in Listing 2.8.
this hypothesis is not pursued in this dissertation, it may be an interesting question for future exploration.

Querying

A dataspace system such as DSToolkit builds on automation to derive mappings automatically, utilising different kinds of associations derived using matching techniques or inferred in the form of more expressive schematic correspondences. However, both processes are likely to give rise to multiple candidate mappings that provide different ways of populating the integration schema. The generated candidate mappings are likely to produce overlapping, inaccurate, incomplete or empty results and therefore feedback is important in perfecting the integration constructs. In the case study, assume that the user poses the following query (expressed in SMql) over the integration schema, denoted by $q_{s\text{int}}$.

In DSToolkit, the user query is translated into an SMql algebraic expression and is then unfolded [Halevy, 2001] using the relevant mappings. Details on how this is done is beyond the scope of this case study. Figure 2.6 shows a subset of the results produced by DSToolkit’s query evaluator over the RDF data sources using a subset of the candidate mappings to populate the integrations schema. This is done in line 8 of Listing 2.6, where the set of mappings $MP_{S_s \rightarrow s_{int}}$ is used to translate the query into sub-queries over the set of source schemas. The result of this process is a set of tuples denoted by $R_{q_{s\text{int}}}$.

Improvement using Feedback

A pay-as-you-go style of integration provides best-effort query answering capabilities that build on automatic techniques to bootstrap a speculative integration. At the same time, the early usage of the integration offers potential for improvement seeking additional information from the user. Feedback provided by the users in the light of usage is a form of payment which allows the system to improve the integration, e.g., by selecting or refining the derived mappings. Over time, the
idea is that the system will respond to user feedback, thus improving the integration and compensating for the usage of automation during the bootstrapping phase.

As an example of a form of feedback that can be utilised, DSToolkit collects feedback on query results. In DSToolkit, feedback takes the form of annotations to query results (line 10 of Listing 2.6); e.g., which tuples from the result set were expected (true positives), which tuples where not expected (false positives) and which tuples were expected but not returned (false negatives). This kind of tuple-based feedback is illustrated in Figure 2.6, and in turn is used to annotate the mappings that derived the result tuples with estimates of precision and recall (line 11 of Listing 2.6). The idea is that subsequent queries are to be answered (line 13 of Listing 2.6) using mappings that satisfy a certain quality expectation by the user (which is above a certain threshold provided as a parameter, CP, line 12 of Listing 2.6). According to [Belhajjame et al., 2013], this process of annotation enables the selection and refinement of the mappings.

<table>
<thead>
<tr>
<th>title</th>
<th>maker</th>
<th>date_created</th>
<th>track_title</th>
<th>expected</th>
<th>not expected</th>
<th>mappings</th>
</tr>
</thead>
<tbody>
<tr>
<td>J.S. Bach French Suites</td>
<td>artist:kerf</td>
<td></td>
<td>Suite No 5 in G major BWV 816: Gigue</td>
<td>✓</td>
<td></td>
<td>mp_1</td>
</tr>
<tr>
<td>J.S. Bach French Suites</td>
<td>artist:kerf</td>
<td></td>
<td>Suite No 2 in C minor BWV 813: Gigue</td>
<td>✓</td>
<td></td>
<td>mp_1</td>
</tr>
<tr>
<td>Phantoms</td>
<td>artist:christian</td>
<td></td>
<td>The Sacrifice</td>
<td>✓</td>
<td></td>
<td>mp_1</td>
</tr>
<tr>
<td>Tempest</td>
<td>artist:4013</td>
<td>2007-07-23</td>
<td>A sea Change</td>
<td>✓</td>
<td></td>
<td>mp_2</td>
</tr>
<tr>
<td>Tempest</td>
<td>artist:4013</td>
<td>2007-07-23</td>
<td>Storm Warning</td>
<td>✓</td>
<td></td>
<td>mp_2</td>
</tr>
<tr>
<td>Suite No 3 in B minor BWV 814: Anglia</td>
<td>artist:kerf</td>
<td></td>
<td>Suite No 3 in B minor BWV 814: Anglia</td>
<td>✓</td>
<td></td>
<td>mp_2</td>
</tr>
<tr>
<td>Aftermath</td>
<td>artist:christian</td>
<td></td>
<td>Aftermath</td>
<td>✓</td>
<td></td>
<td>mp_1</td>
</tr>
</tbody>
</table>

Figure 2.6: Result tuples annotated with user feedback.

For example, the precision of $mp_5$ is calculated as, $tp/(tp + fp)$, where $tp$ is the number of tuples annotated as expected by the user for the specific mapping, and $fp$ is the number of tuples annotated as not expected. Taking into account this information, mapping selection for subsequent queries is able to avoid $mp_5$ which populates the table with identical values for the columns $s_{int}.title$ and $s_{int}.track.title$.

This section has presented a case study that demonstrates how techniques from a DSMS can be used for the integration and querying of RDF data sources. The next section discusses proposals that support uncertainty management throughout the life-cycle of a pay-as-you-go data integration system.
2.6.2 Uncertainty in Dataspaces

Uncertainty management in data integration is a necessity [Halevy et al., 2006b] and has been studied extensively in the literature. Examples include [Dong et al., 2007; Magnani and Montesi, 2007; Sarma et al., 2009; Dong et al., 2009; Sarma et al., 2011; Kuicheu et al., 2013]. This section describes the different sources of uncertainty that can arise in the life-cycle of a dataspace. In addition, we discuss representative proposals for the management of uncertainty as part of the data integration process. Furthermore, whenever possible, this section points out the various sources of uncertainty that arise in LD datasets and contrasts them with those that emerge in data integration systems. Our discussions aim to motivate the work presented in Chapter 4 with regards to reasoning with uncertainty when matching LD datasets.

[Halevy et al., 2006b] state that uncertainty is unavoidable when bringing together data from multiple sources. According to [Franklin et al., 2005], at the core of dataspaces are participants, which are the data sources. Data sources might be in different formats and can be uncertain for several reasons. For example, structured or semi-structured data are often extracted from the Web (e.g., HTML pages) by automatic procedures. Since extraction techniques are neither completely accurate or very robust the data obtained is likely to be uncertain or even inconsistent (e.g., missing information or null values) [Dong et al., 2009; Sarma et al., 2009]. Furthermore, data may be unstructured, i.e., lacking an explicitly defined schema. In such cases, automatic schema extraction techniques are used to infer an approximate structure from the data, which again can be expected to produce uncertain results.

We now relate uncertainty in dataspaces to the WoD. The WoD comprises a great diversity of LD datasets that can be characterised as its participants. According to [Reynolds, 2009] the notion of uncertainty in LD can refer to ambiguity, vagueness and inconsistency. The existing paradigm of LD, with its principles and practices, has no provision for the different types of uncertainty that are likely to arise on the WoD. It also seems that the WoD still suffers from several inconsistencies and errors [Hogan et al., 2010] that are likely to cause various types of uncertainty. Here, we note that several of these sources of uncertainty arise from the very nature of the WoD, which emphasises cross-linking of data from multiple data sources that are often independently developed and maintained by diverse publishers using different publication methodologies and with little or no
coordination.

To exemplify, several approaches exist for mapping data sources adopting a particular data model (e.g., relational) to RDF, e.g., using tools like D2RServer [Bizer and Cyganiak, 2006]. Another interesting case is the DBpedia project [Mendes et al., 2012] which proposed a framework for extracting knowledge from Wikipedia pages and publishing them as LD. With respect to uncertainty, automatic techniques are likely to produce errors, inconsistent data, and missing knowledge thereby causing data from such sources to be unreliable or out of date. Since we cannot assume that the data produced by such processes is correct, a data integration system over LD needs to be able to deal with uncertain data.

From the point of view of the WoD as a Web dataspace, participants (i.e., LD datasets) do not conform to an explicitly defined schema. LD datasets are self-descriptive in the sense that their data is annotated with metadata using terms from various vocabularies that aim to disambiguate their semantics. However, uncertainty can arise from the lack of coordination as to which terms, defined in which vocabulary, are more suitable to annotate a dataset from a specific domain, or else from the lack of knowledge on the suitability of the vocabulary terms used to describe the data in some domains, e.g., bioinformatics. As discussed in Section 2.6, schemas are important to enable a dataspace over LD. To bridge this gap and to enable the automatic initialisation of a dataspace over LD, Chapter 3 introduces a methodology to infer an explicit structure from a given RDF graph. Uncertainty might be represented as the quality (e.g., precision/recall) of the inferred conceptual descriptions from RDF graphs.

A dataspace should also be able to model a set of relationships between participants [Franklin et al., 2005]. More specifically, a dataspace describes a set of (initially inaccurate) schema mappings, which specify the semantic relationships that enable query reformulation among the participants. In a dataspace, automatic techniques are used to bootstrap an initial integration with mappings that are potentially uncertain [Sarma et al., 2009], because, among other reasons, they take uncertain matches as input. Matches are likely to be uncertain due to insufficient knowledge or due to the robustness of the matching techniques. In the case of matches, uncertainty is modelled as a number (typically a similarity score) that serves as a preference ranking mechanism. The possibility of representing match uncertainty as probabilities has been raised in [Sarma et al.,
2009, 2011], but the authors simply assumed that such confidence values or their probability distributions are given in advance, whereas, in contrast, in Chapter 4 we contribute a detailed procedure for deriving such distributions.

Because of this uncertainty in matches, the derived schema mappings are expected to be uncertain as well. Uncertainty for schema mappings is likely to be modelled with rank orderings based on a numeric value, or a weight, that indicates their degree of preference. Alternatively, for schema mappings, uncertainty can be represented with annotations of precision/recall measures, as described in [Belhajjame et al., 2013]. In the description of our case study in Section 2.6.1, we mentioned that schematic correspondences are also derived from matches. For the same reason, schematic correspondences can also be expected to be uncertain. Guo et al. [2013] uses an objective function to rank the most preferable schematic correspondences according to a similarity score that models their level of confidence. Similarly to matches, this confidence measure is used to model uncertainty.

As already discussed, the integration aspect between participants in dataspaces is captured by a set of schema mappings, whereas in the WoD, this aspect is captured by RDF triples (i.e., RDF links). On the WoD, RDF links are published by different processes (often automatic or semi-automatic) at both the instance and conceptual levels. Uncertainty can arise from both instance links (expressed as owl:sameAs statements) and from vocabulary-links, at the conceptual level (e.g., expressed as owl:equivalentClass statements) or even from hidden assumptions in the publication process. Approaches for deriving such links for the WoD (see Section 2.5.2) tend to build on techniques similar to matching, e.g., utilising knowledge from strings or ontologies [Gal and Shvaiko, 2009]. For similar reasons to schema matching, such techniques tend to return uncertain results due to lack of knowledge or because of poor performance of the matching algorithms. It is standard practice for such linkage techniques to associate a confidence score (e.g., similarity score) to the matches.

Another source of uncertainty in data integration is the mediated schema. This may be due to uncertainty introduced during the design process, e.g., designers may not be familiar with the domain at hand. Also, the mappings between the data sources and the mediated schema are sometimes approximate [Sarma et al., 2011; Kuicheu et al., 2013]. Furthermore, in the case of dataspaces, where the mediated schema is inferred automatically by consulting a set of weighted
semantic mappings, approximation is the norm. As a consequence of this inherent uncertainty, the integrated resource may be also uncertain, which then reflects on the quality \textit{query answering}. In the life-cycle of a dataspace, the system leverage user \textit{feedback} to improve. However, users are humans and therefore can also be a source of uncertainty [Jeffery et al., 2008], e.g., by giving inaccurate answers, which impact on the overall quality of the integration. It seems that uncertainty propagates throughout the data integration processes and can have an impact on every stage of the life-cycle of a dataspace [Hedeler et al., 2010b].

Magnani and Montesi [2007, 2010], in research on the management of uncertainty in data integration systems, elaborate on the fact that uncertainty could be valuable when used as a source of evidence, thereby supporting a pay-as-you-go data integration approach. Several specific tasks in the data integration processes (e.g., matching) focus on removing uncertainty at some point of the process, though a choice of the most likely accurate outcomes using filtering strategies [Bernstein et al., 2011], such as thresholds. In reality, uncertainty is not removed from the process but rather reduced, and filtering low-ranked matches could result in the loss of useful or correct information.

In complementary research, Sarma et al. [2011] pointed out that apart from the improvements needed in bootstrapping, a dataspace system needs to evolve the quality of the integration in a pay-as-you-go fashion and that modelling uncertainty at its core could help to improve the data integration system over time. More specifically, it is believed that modelling uncertainty in a dataspace helps to point out where human feedback could be most effective; Section 5.2 points out that this is a potential future direction that can build on the contributions presented in this dissertation.

[Magnani and Montesi, 2007], proposes a generic probabilistic data integration process for dealing with uncertainty throughout the different phases of the process. To identify critical points of their proposed data integration process, they focus on the matching stage, and point out a set of challenges: (i) how are the probabilities produced from matches? (ii) how are probabilities resulted by different matchers to be aggregated? and (iii) how much probability is to be assigned for the match/not-match hypotheses? The contributions described in Chapter 4 provide solutions to these challenges in the LD context. To deal with the management of uncertainty in the context of data integration, [Dong et al., 2007, 2009; Sarma et al., 2009] consider the use of probabilistic schema mappings.
and probabilistic mediated schemas as part of the data integration process and describe how query answering is possible in this setting.

Our brief review on uncertainty management for data integration revealed that most proposals focus on how uncertain schema mappings and mediated schemas from such mappings are represented or semantically defined, and on how query evaluation is performed in their respect. For a pay-as-you-go data integration system over LD, we anticipate the following classes of challenges for uncertainty management:

- **Modelling uncertainty**: As discussed previously, uncertainty in dataspaces can arise at different levels. It seems that it is desirable to model uncertainty throughout the life-cycle in a uniform and principled fashion so that, uncertainty or the decisions made to reduce it, are propagated to other phases of the life-cycle. In other words, how can uncertainty be quantified?

- **Update of the degree of uncertainty**: In the light of new evidence, the degree of uncertainty needs to be updated in a principled fashion. We expect that different sources of evidence can arise from different phases in the life-cycle, for example, evidence from different matchers or evidence from user feedback provided by the user, as part of the improvement phase. In addition, LD sources are likely to be annotated with rich meta data that could perhaps be leveraged as additional sources of evidence.

## 2.7 Discussion and Conclusions

This chapter has highlighted the fact that pay-as-you-go data integration techniques and principles might be suitable for supporting the publication and consumption of heterogeneous LD datasets on the WoD. Research proposals, such as Hermes [Tran et al., 2009] have also recognised the space of opportunities for applying pay-as-you-go data integration over LD. In Hermes, keyword queries are translated into complex SPARQL queries in a setting where integration builds on automatic techniques for deriving semantic mappings between the schema graphs that are constructed over the data sources. Although, Hermes aligns with the pay-as-you-go notion of automatically initialising an integration system, it is lacking in terms of the improvement phase; contrastingly, approaches such as Sig.ma [Tummarello et al., 2010] offers some form of improvement via feedback.
provided by users in terms of selecting which sources should participate results to suggested entity profiles.

In addition to the above, [Franklin et al., 2005] mentioned that dataspaces could be further enhanced by considering recent developments and techniques from other fields. More specifically, from the field of knowledge representation, where ontologies can perhaps leveraged as rich knowledge bases for disambiguating heterogeneous collections of data in dataspaces. In addition, the global identifiers (i.e., URIs) of the Semantic Web could be used as a mechanism for referring to global constants on which some agreement (in terms of structure or semantics) exists among several publishers. Furthermore, the very vision of dataspaces, where the system should be able to offer useful services with minimum or no up-front costs, implies that some degree of uncertainty will arise at various phases of the life-cycle. As discussed in Section 2.6.2, there is a challenge on how this uncertainty is modelled in a way that enables a principled treatment or propagation throughout the life-cycle phases on the benefit of the overall system. The above observation motivated the contribution emerging from objective O3 in Chapter 1. This is discussed in more detail in Chapter 4.

To conclude, this chapter highlighted several opportunities from pay-as-you-go data integration techniques and principles that have not been fully realised. In this dissertation, we propose techniques that support the pay-as-you-go style of integration offered by systems such as DSToolkit. The following chapter discusses a technique that builds on clustering to infer conceptual descriptions over RDF data sources. Chapter 4 describes a principled methodology for quantifying uncertainty based on a subjective probabilistic model and then reasoning with uncertain knowledge, using a Bayesian inference technique known as *Bayesian updating*.
Chapter 3
Structure Inference for Linked Data Sources

“Knowledge of what is does not open the door directly to what should be.”

Albert Einstein, 1939.

LD principles offer a set of simple guidelines for publishing and linking data as RDF triples, however, there is a great deal of flexibility with respect to the terminologies used to describe the data, or how such terminologies are structured. Publishers not only have the flexibility to choose different LD vocabularies, but also to prefer alternative terms in a given vocabulary to facilitate their modelling tasks. Indeed, LD datasets do not need organise data so as to conform to a specific structure analogous to a database schema: instead, data can be structured according to different LD vocabularies. This chapter contributes a methodology for automatically inferring conceptual descriptions for data sources that organise instance level data as RDF triples. The developed methodology identifies recurring structural patterns that are taken to be indicative of conceptual types by partitioning the space of explicitly-stated RDF instances into groups that contain similar instances through the use of a hierarchical clustering technique.

The remainder of this chapter is organised as follows. Section 3.1 briefly introduces cluster analysis, its terminology and its components, as well as, a discussion of the challenges involved in adapting hierarchical clustering algorithm for the goal in hand. An abstract overview of the schema inference methodology that implements the INFERSHEMA operator introduced in Section 2.6.1 is described in Section 3.2. Section 3.3 describes the approach in detail, using a running
example to illustrate the discussions. Section 3.4 presents an experimental evaluation of the approach over different RDF data sources and different scenarios. The results provide empirical evidence that our methodology can infer structural, conceptual descriptions of good quality. Related work regarding this contribution is discussed in Section 3.5. Finally, the chapter concludes in Section 3.6.

3.1 An Overview of Clustering Techniques

Clustering (or cluster analysis) [Jain et al., 1999] is an unsupervised data analysis technique. Given a set of data instances (a.k.a. individuals), cluster analysis is the task of discovering cohesive groups among the data, and categorising the instances into groups called clusters. The goal is to have individuals in a cluster with as high as possible a similarity (or homogeneity) between other individuals in the same cluster, whilst ensuring that the similarity with individuals in other clusters is as low as possible. Similarity is typically a quantification of the degree of commonality in the characteristic features of the data instances.

Typical clustering algorithms utilise information that is present in the given data instances, without appealing to a priori knowledge as to which groups they may form. In this respect, clustering algorithms can be contrasted with classification techniques [Phyu, 2009], in which the groups into which the data instances must be partitioned are known in advance.

The grouping process of a clustering algorithm relies on the existence of a suitable metric that quantifies the similarity between data instances. Such a metric is frequently referred to as a proximity measure [Jain et al., 1999]. Typically, clustering is an activity that involves:

(i) a procedure for data instance selection/extraction, i.e., identifying the features that make an individual eligible for clustering;

(ii) a proximity measure, i.e., a metric that determines the extent to which a pair of individuals are similar. Broadly speaking, such metrics can be of two types: distance measures, which are used to characterise how close or apart two individuals are (e.g., Euclidean distance computed over pairwise attribute values), and similarity measures, which aim to quantify conceptual similarity (e.g., the Jaccard similarity coefficient on the extents of two concepts). The reader is referred to [Rokach, 2010] for more details on similarity and dissimilarity measures;
(iii) a grouping step, i.e., an approach used for partitioning the data into groups. The output of this step is a set of clusters (a.k.a. a clustering). The output clustering is typically characterised by the memberships of each cluster in it. More specifically, the final clustering can be hard (by partitioning data into a number of disjoint clusters) or fuzzy (by allowing a degree of membership of an individual across more than one cluster) [Jain et al., 1999]. For example, partitional (or flat) clustering algorithms, such as \( k \)-means [Macqueen, 1967], group individuals into a single, flat partition where the clusters (of which there is often a predefined number) stand in no explicit relation or hierarchy.

In contrast, hierarchical clustering algorithms produce a sequence of nested partitions, as opposed to a single partition, with clusters sharing individuals with other clusters. Because of this, hierarchical algorithms must use a criterion for merging (agglomerative methods) or splitting (divisive methods) clusters, based on similarity. Typically, the results of a hierarchical approach to clustering are visualised as a dendrogram. Choosing the level of the dendrogram (see an example in Figure 3.4) that satisfies some criteria, allows one to partition the space of individuals into the appropriate clusters.

For the grouping step, alternative techniques exist based on probabilistic (e.g., [Spytkowski and Kwasnicka, 2012]) or on graph-theoretic methods (e.g., [Grygorash et al., 2006]). For an examination of the different approaches for performing clustering, the reader is referred to [Jain et al., 1999; Rokach, 2010; Han and Kamber, 2011].

The technique contributed in this chapter for inferring conceptual descriptions of RDF data sources builds on a hierarchical agglomerative clustering algorithm. For the purposes of facilitating our discussion of the specific details of our contribution, we introduce an abstract description of hierarchical clustering algorithms that uses the agglomerative style. Such an approach to clustering comprises the following steps:

1. Assign each object to a single cluster (referred to as singleton clusters).

2. Compute the similarity between every pair of clusters\(^1\).

\(^1\)In terms of implementation, it is often the case that pairwise similarities are stored in a similarity matrix, in which the \(ij\)-th entry holds the similarity between the \(i\)-th and \(j\)-th cluster.
3. Find the most similar (closest) clusters and merge them into one.

4. Update the similarity of the clusters that were involved in the merging step.

5. Repeat steps (3) and (4) until only one cluster remains, or until a stopping criterion is met.

Several variations of the abstract description above exist due to different implementation choices, such as: (i) the stopping criteria; examples include: a particular number of clusters has been reached, a maximum number of individuals per cluster has been reached, or the maximum similarity for merging (i.e., step 3) is below a certain threshold [Han and Kamber, 2011]; and (ii) the way in which the similarity between the clusters is determined (i.e., the choice of a linkage scheme) [Jain et al., 1999].

In the contributed methodology for schema inference we have considered the following linkage schemes [Rokach, 2010]: single-linkage (where the similarity between clusters is the maximal similarity between elements of each cluster), complete-linkage (where the similarity between clusters is the minimum similarity between elements of each cluster) and average-linkage (where the similarity between clusters is the mean similarity between elements of each cluster) in order to determine the similarity between the clusters. Formal descriptions for each linkage scheme are given later in Section 3.3.3. As regards the termination condition, the contributed clustering algorithm (see Section 3.3.3) terminates when the similarity of the most similar pair of clusters to be merged is below a certain threshold (denoted by $t$). Section 3.4 discusses the empirical choice of the termination threshold, $t \in [0, 1]$, as well as the choice of linkage scheme.

It is recognised that hierarchical clustering approaches produce high-quality clusterings, but often fail to scale well, with a time complexity of at least $O(N^2)$ for clustering $N$ individuals [Jain et al., 1999; Rokach, 2010]. In addition, hierarchical approaches are greedy, since a merging of the clusters cannot be undone (i.e., backtracked). Despite their disadvantages, hierarchical clustering algorithms possess several advantages, such as: the clustering is performed in a deterministic way, they do not require any prior knowledge as to the number of clusters that there are and they produce multiple nested partitions, allowing users to choose their preferred level of partitioning in the dendrogram etc. [Rokach, 2010].

Having discussed some of the advantages offered by a hierarchical clustering approach, this section now briefly elaborates on its choice for the foundation
CHAPTER 3. STRUCTURE INFERENCE FOR LD SOURCES

of our schema inference methodology. The discussion begins by correlating the characteristics of this technique with the improvement philosophy of a dataspace. The sequence of nested partitions discovered by a single run of a hierarchical clustering algorithm offers potential for utilising user feedback to choose the final clustering, according to the user’s requirements (this is discussed as a possible future direction in Chapter 5). This is in contrast to techniques, such as k-means, where the output result of a clustering is flat, and where the outcome depends on many configuration parameters, e.g., a predefined number of clusters, or initial cluster centres [Rokach, 2010; Oikonomakou and Vazirgiannis, 2010]. This means that, in order to obtain a variant clustering result, the algorithm needs to be run several times. According to [Rokach, 2010; Oikonomakou and Vazirgiannis, 2010], hierarchical clustering techniques can handle noise appropriately, provide good clusters often, and require fewer configuration parameters than other approaches. Such characteristics make hierarchical techniques suitable for our case, despite the risk of poor performance which can be mitigated by taking a stratified sample of individuals.

More importantly, and to support the ethos of automatic initialisation (as envisioned by a dataspace), the proposed schema inference step should minimize human intervention as much as possible, yet simultaneously be able to produce a good conceptual description (i.e., a schema) that can facilitate downstream tasks, such as semantic matching, mapping and query rewriting (discussed in Section 2.6.1). This requirement means that the structure inference approach should be as autonomous as possible, therefore the technique should be capable of determining the appropriate number of clusters for a given dataset autonomously (with no upfront costs in terms of configuration). The following section introduces a technique for doing so.

Silhouette Coefficient

As previously mentioned, a hierarchical clustering approach – *agglomerative* (or bottom-up) or *divisive* (or top-down) – produces several clusterings. Kaufman and Rousseeuw [1990] point out that one important parameter in cluster analysis is the number of clusters expected in the data. In partitional techniques this value, denoted by $k$, needs to be determined in advance. In hierarchical techniques, such a value is not required for the technique to run. However, this is somewhat equivalent to choosing the appropriate level of similarity (or dissimilarity) to cut
CHAPTER 3. STRUCTURE INFERENCE FOR LD SOURCES

the final dendrogram. Deciding which clustering best fits the data is a well-known issue in cluster analysis [Kaufman and Rousseeuw, 1990; Halkidi et al., 2001].

Numerous unsupervised clustering evaluation measures exist in the literature [Kaufman and Rousseeuw, 1990; Halkidi et al., 2001; Han and Kamber, 2011] that can be used to determine or approximate the correct number of clusters. Usually, such approaches build on techniques for assessing the quality of the output clusterings. Such techniques typically determine this based on (i) compactness, i.e., how close the elements of a cluster are, and (ii) separation, i.e., how distinct a cluster is from other clusters.

To keep the schema inference methodology independent of any external knowledge, such as the number of clusters, we use a silhouette coefficient [Kaufman and Rousseeuw, 1990] as a measure to determine the most appropriate clustering. Thus, it also, implicitly, determines the number of clusters.

Given a set of individuals, a silhouette coefficient is calculated for each individual. It is a measure of how similar the individual is to other individuals in the cluster to which it belongs compared with individuals from other clusters. Although the silhouette coefficient makes a subjective decision, empirical evaluation (see Section 3.4) showed that clusterings determined using silhouette coefficients correspond to conceptual descriptions of good quality. The mathematical form of the silhouette coefficient is introduced below. A procedure for deriving silhouette coefficients, given a set of individuals, and a clustering is shown in Algorithm 3 (later in this chapter).

Based on the ideas of compactness and separation, a silhouette coefficient is defined as follows [Kaufman and Rousseeuw, 1990]:

\[
sil(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \in [-1, 1], \text{ with } (3.1)
\]

\[
b(i) = \min_k \{B(i, k)\}, \quad (3.2)
\]

where \(i\) is an individual in some cluster, \(a(i)\) is the average dissimilarity between \(i\) and all other individuals in its cluster, and \(b(i)\) is the minimum average dissimilarity between \(i\) and individuals in another cluster \(k\), denoted by \(B(i, k)\). For singleton clusters we follow the definition by Kaufman and Rousseeuw [1990] and assign \(sil(i) = 0\). A silhouette coefficient can take any value in the range \([-1, 1]\), where: values closer to 1 indicate that the distance of an individual from
neighbouring clusters is very high; closer to 0 indicates individuals that are not distinctly in one cluster or another and finally values closer to -1 indicate individuals potentially assigned to the wrong clusters [Kaufman and Rousseeuw, 1990]. It is desirable for the values of the silhouette for each individual to be positive \(a_i < b_i\), and for \(a_i\) to be closer to 0 as possible, since the coefficient will then be closer to \(v_{iz} = 1\).

Having calculated the silhouette values for each individual, the \textit{average silhouette width} (ASW) can be computed for each cluster (denoted by \(ASW_{\text{cluster}}\)), as the mean value of \(sil(i)\), for all \(i \in k\). An overall measure of goodness (denoted by \(ASW_{\text{overall}}\)) can also be obtained for a clustering as the mean of all individual \(sil(i)\) silhouettes. This is defined as:

\[
ASW_{\text{overall}}(k) = \frac{\sum_{i=1}^{n} sil(i)}{n},
\]

where \(n\) denotes the total number of individuals. To determine the best clustering, and thus decide on the number of clusters, the clustering algorithm (see Algorithm 2, later in this chapter) chooses the clustering with the highest \(ASW_{\text{overall}}\). The derivation of the individual silhouettes for each clustering has \(O(N^2)\) complexity [Kaufman and Rousseeuw, 1990].

3.2 An Overview of the Contributed Approach

As previously discussed, data integration involves obtaining an understanding of the relationships between the data sources and the mediated schema. Such an understanding typically builds on information from the conceptual descriptions of the data sources to be integrated. In the context of LD, although publishers are encouraged to use good publication guidelines, such as the recurring use of terminologies from well known vocabularies [Hyland et al., 2014], there will inevitably be terminologies from overlapping domains or publishers that combine them in different ways when publishing their datasets. LD datasets can therefore be characterised as schema-less, as the data do not strictly obey a predefined conceptual structure and instead, often alludes to multiple vocabularies.

The case study from Section 2.6.1 motivated the need of a technique for inferring conceptual descriptions over RDF data sources. The derived conceptual descriptions should determine how concepts in a source are described and how
they are organised, simultaneously enabling a dataspace system to bootstrap an integration over RDF data sources. This is one of the objectives of this dissertation (see Section 1.4). To support the ethos of a bootstrapped system, a dataspace should be able to infer such conceptual descriptions automatically. This section presents an overview of the developed methodology. Subsequent sections describe in more detail the INFERSHEMA operator introduced in Listing 2.6.

An abstract description of the methodology for inferring conceptual descriptions over RDF data sources is depicted in Figure 3.1. Given an RDF data source or a SPARQL dataset (Stage 1), the approach builds on the assumption that recurring structural patterns of a possible concept can be discovered by observing RDF instance level data. To detect such patterns, the approach distinguishes RDF triples that describe instance data (Stage 2) and extracts a representation of data instances (i.e., individuals) that is suitable for cluster analysis (Stage 3). It is worth noting at this point that traditional clustering methods, such as the hierarchical clustering described in Section 3.1, do not work over (potentially large) interconnected RDF graphs as they typically collect data about a great many distinct individuals. Thus, for the application of a clustering technique (Stage 4) over an RDF graph, one needs to consider the following challenges:

C.1 What exactly constitutes a suitable individual for clustering in an RDF graph?

C.2 How are such individuals extracted and represented?

C.3 Given a pair of individuals from an RDF graph, how is their similarity (or distance) computed?

C.4 How is an appropriate number of clusters determined?

Before continuing with a detailed description of the developed methodology, we briefly introduce some concepts required by what follows in this chapter.

**Definition 13 (Class usage).** Given an RDF resource, we call class usage the set $CU = \{c_1, \ldots, c_n\}$, where $c_i, i = 1, \ldots, n$, is an RDF term that appears in the o position of an RDF triple when $p$ is an `rdf:type` and o is neither a `meta-class` nor a `meta-property`. 
CHAPTER 3. STRUCTURE INference FOR LD SOURCES

Figure 3.1: Abstract description of schema inference methodology.

**Definition 14** (*Predicate usage*). Given an RDF resource, we call *predicate usage* the set $PU = \{p_1, \ldots, p_n\}$, where $p_i = \langle\text{term}, \text{count}\rangle$, $i = 1, \ldots, n$, where *term* is the RDF term that appears in the $p$ position of an RDF triple when $p$ is not a *meta-property*, represented by its CURIE name as prefix:localName, and *count* is an integer number representing the number of occurrences of each RDF term in the resource.

**Definition 15** (*Candidate Description*). Given an RDF resource, a *candidate description* is the union of the elements of the *class usage* and *predicate usage* sets, $CU \cup PU = \{e : e \in CU \text{ or } e \in PU\}$.

Using cluster analysis, if homogeneous RDF instance level data can be organised into clusters, then data instances can be utilised to heuristically discover evidence of recurring structural patterns thereby revealing the resource’s conceptual structure. This evidence is used to annotate the clusters (*Stage 5*) and finally to infer a conceptual summary over the RDF graph that captures the *classes* that the data instantiate, their *properties* and their relationships to other classes (*Stage 6*). In the context of this dissertation, we call *schema* (or *conceptual summary*) *inference* the process of abstracting RDF data instances into a representation that is a subset of the Entity-Relationship (ER) data model constructs [Elmasri et al., 1985]. The ER model is rich enough: (i) to capture the
conceptual structure of an RDF graph along with constraints that assist a dataspace in deriving the relationships (i.e., mappings) required for source integration; and (ii) to enable complex SPARQL graph patterns to be formulated over RDF graphs.

3.3 Schema Inference using Clustering

This section discusses the details of the developed methodology for inferring conceptual descriptions given an RDF graph. To guide the reader, a sub-graph from the Jamendo RDF graph is used as a running example (depicted in Figure 3.2).

3.3.1 Identify Input RDF Graph

As with the abstract description, our methodology for schema inference is presented with an RDF graph either in the form of a single RDF set of triples retrieved from the WoD, or the location of a SPARQL dataset. In the former case, the RDF graph is stored in a local triple-store. In the later case, an RDF graph is retrieved via the URI of the query interface that provides access to the SPARQL dataset. It is worth noting that a SPARQL dataset usually comprises several named graphs (as defined in Section 2.4); thus, the user needs to specify which of the named graphs is to be used as input to the schema inference technique.

The methodology assumes that sufficient resources and bandwidth exist to store and retrieve an RDF graph of any size and provides a simple algorithm for restricting the initial RDF graph to a manageable size if needed. In so doing, our methodology retrieves a finite set of RDF resources that together comprise a sub-graph of the initial RDF graph selected at random (see Algorithm 1).

The algorithm uses a SPARQL SELECT query to discover the URIs of classes (as of Definition 4) that might exist in the initial RDF graph (line 1 of Algorithm 1). For each of the URIs of classes discovered, the algorithm performs a random selection over the subject URIs that are typed (lines 2-3 of Algorithm 1) with such URIs. Using the randomly selected typed subject URIs, a SPARQL DESCRIBE query is posed over the original RDF graph to retrieve all the RDF triples used to describe the subject URI. The results from the query are stored

\(^2\)A back-ended database for storing and querying RDF graphs. In terms of implementation, we have used the Jena TDB RDF store, http://jena.sourceforge.net/TDB/
Figure 3.2: Interconnected RDF sub-graph extracted from the Jamendo dataset.
Algorithm 1 Create sub-graph from initial RDF graph.

Require: Single RDF dump or SPARQL dataset URI

1. $C \leftarrow \text{SELECT DISTINCT } ?\text{class} \text{ FROM NAMED } <G_{\text{uri}}>$
   $\text{WHERE } \{ \text{GRAPH } g \{ ?s \text{ rdf:type } ?\text{class} \} \}$

2. for each class URI $c \in C$ do
   $\triangleright$ find URIs of classes using rdf:type
   3. $S_{\text{typed}} \leftarrow \text{select at random } N \text{ number of distinct subject URIs}$
      where the value of rdf:type is $c$
   4. for each typed subject URI $s \in S_{\text{typed}}$ do
      5. $G_e\text{.add()} = \text{DESCRIBE } <s> \text{ FROM NAMED } <G_{\text{uri}}>$
         $\text{WHERE } \{ \text{GRAPH } g \{ ?s \text{ ?p } ?o \} \}$
   end for
   6. end for

7. $U \leftarrow \text{SELECT DISTINCT } ?s \text{ FROM NAMED } <G_{\text{uri}}>$
   $\text{WHERE } \{ \text{GRAPH } g \{ ?s \text{ ?p } ?o \}$
   $\text{OPTIONAL } \{ ?s \text{ rdf:type } ?\text{class} \}$
   $\text{FILTER } (\text{!BOUND } (?\text{class})) \} \}$

8. $S_{\text{untyped}} \leftarrow \text{select at random } N \text{ number of untyped subject URIs}$
    from $U$

9. for each untyped subject URI $s \in U$ do
   10. $G_e\text{.add()} = \text{DESCRIBE } <s> \text{ FROM NAMED } <G_{\text{uri}}>$
        $\text{WHERE } \{ \text{GRAPH } g \{ ?s \text{ ?p } ?o \} \}$
   end for

11. return $G_e$

as part of a new RDF graph, denoted by $G_e$ (lines 4-7 of Algorithm 1). The RDF model does not impose any restrictions on the use of rdf:type statements to determine whether a certain resource is an instantiation of some class. Such resources (referred here as untyped resources) are potentially useful for understanding the structure of a given RDF graph; therefore, the resultant sub-graph should also capture such RDF resources. In so doing, the algorithm uses a SPARQL SELECT query to discover the URIs of all such resources (line 8 of Algorithm 1). Then, and by randomly selecting a predefined number of such untyped resources, it uses a SPARQL DESCRIBE query to retrieve all the RDF triples used to describe each selected resource and to add them to $G_e$. The result of this process is an RDF graph which is a subset of the initial graph created using random sampling.
We recognise that the pruning phase of randomly selecting URI resources can lead to insufficient knowledge regarding the structure of the original RDF graph, however, empirical evaluation showed that the sub-graphs extracted contain enough characteristics that can be utilised for eliciting the structure of a given RDF source.

3.3.2 Pre-processing

RDF Instance Extraction

Having obtained the RDF graph, the schema inference methodology proceeds by extracting from the RDF graph suitable RDF instances for cluster analysis. Previously, we referred to this challenge as C.1. Here, by RDF instance extraction, we mean the RDF triples that comprise a relevant sub-graph that describe a given resource. In an RDF graph, data instances are interconnected through RDF links that are explicitly specified. As described in Section 2.2.4, LD advocates the use of external RDF links to enable the discovery of new knowledge. In addition, RDF links can be internal within a single LD dataset (i.e., the subject and object URIs of an internal RDF link are from the same namespace [Heath and Bizer, 2011]).

The question arises as to what constitutes enough information for an RDF resource when an RDF graph is potentially interconnected with both internal as well as external RDF links. To understand this, we consider the RDF resource jar:337090 that describes a music artist. This RDF resource has internal RDF links (e.g., foaf:maker) to a resource that provides information about a recording. Moreover, jar:337090 has external RDF links (e.g., foaf:based_near) to a resource from a different namespace, that describe information about the location of the music artist. If we consider all the RDF links for jar:337090 we are likely to retrieve a very large RDF graph. Thus, the challenge is to find a reasonable subset of the graph that describes each RDF resource.

For our purposes, this challenge was regarded as being beyond the scope of this dissertation. Different approaches proposed as solutions to this challenge are described in [Grimnes et al., 2008]. Our schema inference methodology considers a suitable representation of the sub-graph comprising all the immediate predicates used to describe the given resource, without dereferencing any external RDF links.

The highlighted RDF triples from Figure 3.2 indicate the kinds of sub-graphs
extracted to represent an individual in a given RDF resource. The following section discusses how knowledge from such sub-graphs is used to assign to each RDF resource a representation that enables their similarities to be computed using a similarity measure (previously, we referred to this challenge as C.2), which is a crucial requirement for performing cluster analysis, as discussed in Section 3.1.

Representing Individuals

In the contributed schema inference methodology, RDF resources extracted from an RDF graph are represented as candidate descriptions. From now on we refer to the set of candidate descriptions as the population of individuals, denoted by \( CD \). Given an RDF graph, our methodology considers every RDF resource that describes instance level data, extracts its sub-graph (as described previously) and represents it as a candidate description, \( cd_i \in CD \), where \( i = 1, \ldots, n \). A candidate description (see Definition 15), consists of the elements of both the class usage set, denoted by \( CU \), and the predicate usage set, denoted by \( PU \). Consider Figure 3.3, which shows an example of a candidate description representing the sub-graph extracted for the RDF resource, identified by \texttt{jar:337089}:

\[
\begin{align*}
CU_1 &= \{\text{foaf:Person, mo:MusicArtist}\} \\
PU_1 &= \{\langle \text{foaf:name, 1}, \text{mo:biography, 1}, \text{foaf:made, 1} \rangle\}
\end{align*}
\]

\( cd_1 = \{\langle \text{foaf:Person, mo:MusicArtist}, \\
\langle \text{foaf:name, 1}, \\
\langle \text{mo:biography, 1}, \\
\langle \text{foaf:made, 1} \rangle \rangle \}
\]

Figure 3.3: Example of an individual represented as a candidate description.

Once each RDF resource is represented by a candidate description, the schema inference approach proceeds with cluster analysis using a hierarchical (agglomerative) clustering algorithm. Our methodology builds on the assumption that recurring structural patterns of a possible concept are likely to be discovered by observing similar RDF instances organised into homogeneous groups. Thus, cluster analysis is used here to obtain an understanding of how the data is partitioned. In our case, cluster analysis is performed over a pool of individuals (i.e., an abstract description of the actual sub-graph of RDF instances). The
CHAPTER 3. STRUCTURE INFERENCE FOR LD SOURCES

95
goal is for identified groups to inform the identification of classes that the data instantiate, of properties, and of the relationships between classes.

As previously discussed, prior to the grouping step, typical clustering techniques use a metric to determine the degree to which two data instances are similar. In this respect, clustering is commonly performed over data instances represented as numerical vectors. Such vectors are viewed as coordinates in an N-dimensional space; therefore it is trivial to compute their distances using geometric distance functions, such as the Euclidean or Manhattan distance [Jain et al., 1999; Rokach, 2010]. The following section discusses a simple function that can be used to determine the distance between two candidate descriptions as a solution to the challenge referred to as C.3. Then, it proceeds with a description of the clustering algorithm used to find groups within the set of candidate descriptions.

3.3.3 Cluster Candidate Descriptions

To facilitate our discussion, we follow an example of such candidate descriptions (see Listing 3.1), extracted from the Jamendo sub-graph (various of these are shown in Figure 3.2).

From a brief review of the candidate descriptions representing sub-graphs of RDF instances, one can observe that there is a lack of compatibility between resources that are instantiations of the same class. To exemplify, consider the RDF resources jar:337089 and jar:337090, which are instantiations of the class mo:MusicArtist. The arity of their predicates is not the same, nor are the kinds of predicates used to describe each resource. More specifically, jar:337089 suggests that an RDF resource that describes a musician should capture information about the musician’s biography (e.g., mo:biography), whereas jar:337090 suggests that this knowledge is missing from the description of the musician. Furthermore, jar:337090 reveals the existence of other predicates viz., foaf:img that point to a Web URI that holds an image of the musician, foaf:homepage a Web URL which points to the personal web page of the musician, and foaf:based_near reveals information about the musician’s location. Such relationships about the predicates and the structure of the data are implicit in an RDF data source. These inconsistencies give rise to variations in the layout structure of a concept, which creates complications for a schema inference algorithm.
Listing 3.1: Pool of candidate descriptions comprising the set $CD$.

Such complications suggest that evidence of how data is structured can be made more explicit when several of the same RDF instances are organised in the same groups. This then suggests, or not, the existence of a concept, of possible properties that characterise a concept’s structure, as well as of relationships between concepts.

The hierarchical (agglomerative) clustering algorithm used in our schema inference technique is described in Algorithm 2. Commonly, a hierarchical clustering algorithm utilises a data structure known as a similarity matrix, denoted by $M$, to cache the pairwise similarities computed for each pair of candidate descriptions from the pool of individuals. In so doing, the algorithm constructs a similarity matrix $M = |CD| \times |CD|$ from the candidate descriptions $CD$. Given two candidate descriptions $(cd_i, cd_j)$ where $i \geq 1$, $j \leq |CD|$, the algorithm computes their similarity using sets of local names, extracted from each candidate description.
**Definition 16 (Set of local names).** Given a candidate description \( cd \in |CD| \) the set of *local names* is either:

1. the union of the RDF terms that are the elements of the *class usage* and the *term* part of the \( \langle \text{term}, \text{count} \rangle \) of their *predicate usage* sets, if and only if none of the class usage sets are empty, or

2. the union of the RDF terms that are part of the \( \langle \text{term}, \text{count} \rangle \) elements of their *predicate usage* sets, if any of the *class usage* sets is empty.

Let \( \text{createLocal}(\cdot) \) be a function that given a candidate description, e.g., \( cd \), extracts a set of local names (as of Definition 16), denoted by \( L_{cd} \). Considering a pair of candidate descriptions \( (cd_i, cd_j) \), we denote by \( L_{cd_i} \) and \( L_{cd_j} \) the set of local names extracted by \( cd_i \) and \( cd_j \) respectively. As a concrete example, assume that the candidate descriptions \( cd_1 \) and \( cd_2 \) from Listing 3.1 both have a non-empty set of class usage. The derivation of similarities is shown in Listing 3.2. As another example, consider the candidate descriptions \( cd_1 \) and \( cd_3 \). In this case, \( cd_3 = \emptyset \), so computing the similarity between \( (L_{cd_1}, L_{cd_3}) \) is different as shown in Listing 3.2.

---

### Example

- \( cd_1 \): \{Person, MusicArtist\}, \{name, biography, made\}
- \( cd_2 \): \{MusicArtist\}, \{name, based\_near, made, img, homepage\}

\[
\text{cd}\_\text{sim}(cd_1, cd_2) = \frac{|L_{cd_1} \cap L_{cd_2}|}{|L_{cd_1} \cup L_{cd_2}|} = \frac{3}{8} = 0.38
\]

- \( cd_3 \): \{\emptyset\}, \{name, biography, made\}

\[
\text{cd}\_\text{sim}(cd_1, cd_3) = \frac{|L_{cd_1} \cap L_{cd_3}|}{|L_{cd_1} \cup L_{cd_3}|} = \frac{0}{9} = 0.0
\]

---

Listing 3.2: Example of the sets of local names used while computing the similarity between a pair of candidate descriptions.

We define \( \text{cd}\_\text{sim}(\cdot) \) as a function that computes the Jaccard similarity between a pair of candidate descriptions using their extracted local names sets, as follows:

\[
\text{cd}\_\text{sim}(cd_i, cd_j) = \frac{|L_{cd_i} \cap L_{cd_j}|}{|L_{cd_i} \cup L_{cd_j}|} \in [0, 1]. \tag{3.4}
\]
Algorithm 2 Cluster pool of candidate descriptions.

Require: Set of $CD = \{cd_1, cd_2, \ldots, cd_{|CD|}\}$

1: Clustering, $C ← \{\emptyset\}$
2: $m ← 0$
3: $U^m ← \{\{cd_1\}, \{cd_2\}, \ldots, \{cd_{|CD|}\}\}$
4: Construct similarity matrix $M = |CD| \times |CD|$
5: Let $(U^m_i, U^m_j)$ be the most similar pair in $M$:
6: $\argmax_{(U^m_i, U^m_j) \in M} \text{cluster sim}(\{U^m_i\}, \{U^m_j\})$
7: $\max ← \text{cluster sim}(\{U^m_i\}, \{U^m_j\})$
8: while $(\max \geq t)$ do
9: $m ← m + 1$
10: $U^m_{ij} ← U^m_{i(m-1)} \cup U^m_{j(m-1)}$
11: $U^m ← (U^{(m-1)} \setminus \{U^m_{i(m-1)}, U^m_{j(m-1)}\}) \cup U^m_{ij}$
12: Derive silhouette coefficient, using $CD, C,$ and $M$
13: Update similarity matrix $M$
14: Let $(U^m_i, U^m_j)$ be the most similar pair in $M$:
15: $\argmax_{(U^m_i, U^m_j) \in M} \text{cluster sim}(\{U^m_i\}, \{U^m_j\})$
16: $\max ← \text{cluster sim}(\{U^m_i\}, \{U^m_j\})$
17: end while
18: return $C$

Algorithm 3 Calculate silhouette coefficient.

Require: Set of $CD = \{cd_1, cd_2, \ldots, cd_{|CD|}\}$
Require: A clustering of $k$ clusters $C_1, \ldots, C_k$ where $1 \leq i \leq k$
Require: A dissimilarity matrix, $M_d = |CD| \times |CD|$
1: $\text{silCoefMap} : \text{Map}(cd_i \rightarrow \text{Double})$
2: for each $cd_i \in CD$ do
3: $C_i ← \text{find cluster where } cd_i \text{ is an element}$
4: $a(cd_i) = \frac{\sum_{cd_i' \in C_i, cd_i' \neq cd_i} \text{dissim}(cd_i, cd_i')}{|C_i| - 1}$
5: $C_j : 1 \leq j \leq k, j \neq i \left(\sum_{cd_i' \in C_j} \text{dissim}(cd_i, cd_i') \right) / |C_j|$
6: Compute the silhouette coefficient for $cd_i$ as
7: $\text{sil}(cd_i) = \frac{b(i) - a(i)}{\max\{a(cd_i), b(cd_i)\}}$
8: $\text{silCoefMap}.\text{add}(cd_i, \text{sil}(cd_i))$
9: end for
10: return $\text{silCoefMap}$
Having computed the pairwise similarities between candidate descriptions and stored them into a similarity matrix, the clustering algorithm proceeds as follows. Initially, and to keep track of each iteration, the algorithm initialises a sequence $0, 1, \ldots, (m-1)$ (line 1 of Algorithm 2). Then, it assigns each $cd \in CD$ to a singleton cluster. The set of clusters (i.e., the clustering) derived at each iteration is denoted by $U^m$, so $U^0 = \{ \{cd_1\}, \{cd_2\}, \ldots, \{cd_{|CD|}\} \}$ (line 2 of Algorithm 2). To produce a sequence of nested partitions of the space, Algorithm 2 implements a merging step. As described in Section 3.1, this is a characteristic of agglomerative hierarchical methods. Thus, in lines 4-5 the algorithm identifies the most similar pair of clusters (according to the similarity measure used) in order to merge them (line 6 of Algorithm 2). Before the merging step takes place, the termination condition of the algorithm is checked (line 7 of Algorithm 2), and if it does not hold, the algorithm proceeds iteratively by merging the most similar pair of clusters at each step to form the next clustering, $m = m + 1$, where $U^{(m-1)}_i$ and $U^{(m-1)}_j$ until the merging process ends.

As a result of the merge step, at each iteration the number of clusters decreases by one, that is $|U^m| = |U^{(m-1)}| - 1$. The result of the merging step is a new cluster that is $U_{ij}^m = U^{(m-1)}_i \cup U^{(m-1)}_j$ and a new clustering (lines 8-9 of Algorithm 2). The clustering $C$ resulting from the merge step (line 10 of Algorithm 2) is stored to be used later as an input to a procedure that determines its overall average silhouette width ($ASW_{overall}$ from Section 3.1). The algorithm for computing individual silhouette coefficients is shown in Algorithm 3. As stated in Section 3.1, our clustering algorithm calculate silhouettes for each individual in order to determine the best clustering and, thus, an appropriate number of clusters, that is a value for $k$. The next step is for the algorithm to update the similarities of the clusters that were involved in the merging step and other clusters from the similarity matrix (line 11 of Algorithm 2).

In the literature, different linkage schemes have been proposed for deriving the similarity between a pair of clusters [Rokach, 2010]. In our schema inference technique, the similarity between clusters $U^m_i$ and $U^m_j$ is defined to be the mean similarity between the elements of each cluster (a.k.a. the average-linkage
scheme). More formally, average-linkage is defined as:

$$ \text{cluster}_{\text{sim}}(U^m_i, U^m_j) = \frac{\sum_{cda \in U^m_i} \sum_{cd_b \in U^m_j} \text{cd}_{\text{sim}}(cda, cd_b)}{|U^m_i||U^m_j|} \in [0, 1]. \quad (3.5) $$

The behaviour of different linkage schemes has been extensively discussed in the literature. For an extensive discussion on linkage schemes the interested reader is referred to [Jain et al., 1999; Rokach, 2010; Han and Kamber, 2011]. In this chapter, and for the purposes of completeness, the results obtained from using average-linkage were compared with single-linkage and complete-linkage. More formally:

- **Single Linkage:** The similarity between clusters $U^m_i$ and $U^m_j$ is the maximal similarity between elements of each cluster, defined as:

$$ \text{cluster}_{\text{sim}}(U^m_i, U^m_j) = \max_{cda \in U^m_i, cd_b \in U^m_j} \text{cd}_{\text{sim}}(cda, cd_b). \quad (3.6) $$

- **Complete Linkage:** The similarity between clusters $U^m_i$ and $U^m_j$ is the minimum similarity between elements of each cluster, defined as:

$$ \text{cluster}_{\text{sim}}(U^m_i, U^m_j) = \min_{cda \in U^m_i, cd_b \in U^m_j} \text{cd}_{\text{sim}}(cda, cd_b). \quad (3.7) $$

Section 3.4 describes the results of obtaining different clusterings with the use of different linkage schemes, to characterise the similarity between a pair of clusters. The empirical results showed that the above linkage schemes tend to produce similar results. Finally, regarding the termination condition, the clustering algorithm stops when the most similar pair of clusters to participate in the merging step is below a certain threshold, that is $\text{cluster}_{\text{sim}}(U^m_i, U^m_j) < t$.

Hierarchical clustering algorithms produce a dendrogram, i.e., a nested hierarchy of graphs that can be interpreted as different clusterings of the pool of individuals. The dendrogram can be cut at the desired dissimilarity level, forming different partitions of the space (i.e., clusterings). For example consider the dendrogram in Figure 3.4, a cut at 0.6 would result in $k = 4$ clusters, i.e., $\{\{I.10, I.5, I.8\}, \{I.9, I.4, I.7\}, \{I.1, I.6\}, \{I.11, I.2, I.3\}\}$ whereas a cut at 0.4 would yield 5 clusters, further splitting I.9 from I.4 and I.7.
3.3.4 Annotation of Clusters

At the end of the cluster analysis step (Stage 4) in Figure 3.1, the maximum overall average silhouette coefficient is selected to determine the final clustering. The clustering result contains a best-effort partitioning of the population of individuals (which are represented here as candidate descriptions). To derive a structural summary of the data from an RDF graph, the schema inference methodology performs an analysis of each cluster to discover evidence of recurring patterns aimed at revealing the structure that the data adheres to.

Class Annotation

Our approach uses a simple set of metrics to describe the structural characteristics that hold in each cluster. This process is essential for the methodology to be able to derive conclusions on the various schema constructs to be inferred (in the case of the ER model, the entity types, attributes and binary relationships).

The intuition is that sufficiently similar candidate descriptions end up in the same cluster, thus suggesting a conceptual structure for each concept. At this stage, the algorithm looks to annotate each cluster with a candidate class (see Definition 4) that the candidate descriptions (i.e., the members of the cluster)
instantiate. This evidence is subsequently used to infer an entity type, using the ER model notation. For each cluster \( U_i \) of size \( n_i \) with candidate descriptions \( \{cd_1, cd_2, \ldots, cd_n\} \) as elements of the cluster, the algorithm considers the class instantiation, i.e., the number of candidate descriptions from the cluster that are typed as a particular class \( C_i \). In the given RDF graph, this is obtained with a count of the URIs at the object position of an RDF triple, where \( p = \text{rdf:type} \), for each of the candidate descriptions appearing in the cluster. This is derived as follows:

\[
ClassInst(C_i) = |t = (s, p, o)|, \text{ where } p = \text{rdf:type}, \ o = C_i \quad (3.8)
\]

It is worth mentioning that a cluster may contain individuals that all are instantiations of the same class. Instead, it may be that one or more subsets of them belongs to different classes. Finally, it may be that some or all individuals are missing RDF typing information. Considering the above cases, in addition to class instantiation, the approach assigns a confidence value for the RDFS or OWL class suggested by the algorithm, as follows:

\[
ClassConf(C_i) = \frac{\sum_{i=1}^{n} ClassInst(C_i)}{n} \quad (3.9)
\]

As an example, consider the pool of candidate descriptions from Listing 3.1, and assume that \( cd_1 \) to \( cd_3 \) are elements of the same cluster. Listing 3.3 shows an example of the computed class instantiation for the suggested classes \texttt{foaf:Person} and \texttt{mo:MusicArtist} along with their confidence values.

\[
\begin{align*}
ClassInst(\texttt{foaf:Person}) &= 1 \\
ClassConf(\texttt{foaf:Person}) &= \frac{ClassInst(\texttt{foaf:Person})}{ClassInst(\texttt{foaf:Person}) + ClassInst(\texttt{mo:MusicArtist})} = \frac{1}{3} \\
ClassInst(\texttt{mo:MusicArtist}) &= 2 \\
ClassConf(\texttt{mo:MusicArtist}) &= \frac{ClassInst(\texttt{mo:MusicArtist})}{ClassInst(\texttt{foaf:Person}) + ClassInst(\texttt{mo:MusicArtist})} = \frac{2}{3}
\end{align*}
\]

Listing 3.3: Annotate clusters with evidence for classes.

Our approach uses a simple heuristic to select the class that the data is likely to instantiate by selecting the class that has the highest confidence value. The
intuition behind this decision is that the majority of individuals of the most popular class are likely to contain sufficient evidence to infer the structure of a particular concept. Although this could be considered a naive decision criterion, experimental evaluation, as discussed in Section 3.4, reveals it leads to good outcomes. We note that it may be possible to infer more complex relationships from the clusters. For example, when individuals that belong to different classes are grouped in the same cluster, one could consider this as evidence for inferring subsumption or equivalence relations. Such relations could be expressed as entity type hierarchies, or alternatively using OWL axioms as part of a simple ontology e.g., $C_1 \text{rdfs:subClassOf} C_2$ or $C_1 \text{owl:equivalentClass} C_2$. Inferring such relationships from the derived clusters is left for future work.

It is also possible that the types of individuals from a cluster are not made explicit. In such cases, the cluster is annotated with a special unknown label. The approach proceeds to inferring the predicates that describe the structure of an unknown concept, as well as the relationships with other classes. We note that a dataspace may use feedback provided by users to improve the integration. It may be possible to ask users to supply class labels for unknown classes during the improvement phase of a dataspace. Additionally, the inference semantics of RDFS and OWL can be used to perform reasoning to learn the type of an unknown class using the rdfs:domain and rdfs:range inference rules on the predicates, which are used to describe instances of that class; we have left this for future work.

Properties Annotation

At this stage, our algorithm leverages knowledge from the individuals present in each cluster to obtain evidence regarding properties that describe the structure of possible concepts. Here, we mean by properties the literal RDF triples that are used to describe resources. To annotate clusters with evidence regarding properties, our approach considers property instantiation, i.e., the number of distinct times a property $P_j$ appears in the candidate descriptions in the cluster. Given the RDF sub-graph comprising the set of candidate descriptions in a cluster, property instantiation is computed as a count of the RDF triples in which the property is used in the predicate position, as follows:

$$\text{PropInst}(P_j) = |t = (s, p, o)|, \text{ where } p = P_j.$$  (3.10)
Note that, in an RDF graph, there are no guarantees that all RDF instances of a particular class will use the same number of properties, nor of which properties are present. It is therefore desirable to know which attributes are missing and which attributes are single-valued, multi-valued and composite. For example, data from resources that have missing information can be included in the final SPARQL query result with the use of the \texttt{OPTIONAL} keyword. This is beyond the scope of this dissertation. However, it is touched upon here to motivate the need to make the inferred conceptual descriptions as expressive as possible.

To conclude whether an attribute is optional to some of the RDF instances for a particular class, each property is assigned a confidence value that indicates how likely its occurrence is. The confidence value is normalised over the total number of candidate descriptions that exists within a cluster, denoted by $n_i$, as follows:

$$\text{PropConf}(P_j) = \frac{\text{PropInst}(P_j)}{n_i}. \quad (3.11)$$

A confidence value close to 1 indicates strong evidence that the property is used to describe almost all or all the individuals of a particular class. On the other hand, a low confidence value indicates evidence of an optional property. As an example, consider the pool of candidate descriptions from Listing 3.1, again assuming that $cd_1$ to $cd_3$ belong to the same cluster. Listing 3.4 shows the annotations that result.

\[
\begin{align*}
\text{PropConf(foaf:name)} &= \frac{\text{PropInst(foaf:name)}}{3} = \frac{3}{3} \\
\text{PropConf(mo:biography)} &= \frac{\text{PropInst(mo:biography)}}{3} = \frac{1}{3} \\
\text{PropConf(foaf:homepage)} &= \frac{\text{PropInst(foaf:homepage)}}{3} = \frac{2}{3} \\
&\ldots
\end{align*}
\]

Listing 3.4: Annotate clusters with evidence for properties.
CHAPTER 3. STRUCTURE INFERENCE FOR LD SOURCES

Relationships Annotation

Section 2.3 discussed how RDF terms defined in various LD vocabularies are used to describe the data in an RDF graph. In LD vocabularies, resources defined as instances of, e.g., rdf:Property or owl:ObjectProperty are used to express relations that link the instances of the concept defined as the domain (i.e., rdfs:domain) of an RDFS or OWL property definition with instances of the concept defined as the range (i.e., rdfs:range) of a property definition. Such properties are often referred to as object properties. Object properties are used in our approach to annotate clusters with evidence that can lead to potential relationships between concepts.

To annotate clusters with evidence regarding possible relationships between concepts, our approach uses a simple heuristic over object properties: an RDF triple that is an RDF link, rather than a literal RDF triple, is a candidate relationship, and it is a relationship in the RDF graph if it refers to resources that instantiate a different class in the same RDF graph.

The approach explores candidate descriptions from each cluster to discover any object properties that relate to candidate descriptions from different clusters. For each object property discovered, our approach computes the relationship instantiation, i.e., the number of times an object property $OP_j$ appears in candidates descriptions from the cluster. Given the RDF sub-graph comprising the set of all candidate descriptions in a cluster, relationship instantiation is computed as a count of the RDF triples in which the object property is used in predicate position, as follows:

$$\text{RelInst}(OP_j) = |t = (s, p, o)|, \text{where } p = OP_j. \quad (3.12)$$

In addition, for each object property, our approach derives the proportion of candidate descriptions from a cluster that suggests the existence of a relationship to a different cluster. This is expressed as a confidence value (which is later used to suggest a relationship between inferred concepts) computed as:

$$\text{RelConf}(OP_j) = \frac{\text{RelInst}(OP_j)}{n_i}. \quad (3.13)$$

As an example, let us consider the pool of candidate descriptions from Listing 3.1; we assume that the clustering algorithm partitioned the candidate descriptions in the following two clusters $U_1 = \{cd_1, cd_2, cd_3\}$, and $U_2 = \{cd_4, cd_5, cd_6\}$. 
Given the URIs of the resources expressed as candidate descriptions, their object properties are discovered with a SPARQL query, as shown in Listing 3.5.

```
SELECT ?objProp ?dom ?range
WHERE {
  OPTIONAL {jar:337089 a ?dom} .
  OPTIONAL {?s2 a ?range} .
}
```

Listing 3.5: Example of SPARQL query to discover object properties.

Candidate descriptions from cluster $U_1$ represent instantiations of the class `mo:MusicArtist`. An example of an object property discovered from these candidate descriptions is `foaf:made`, which reveals a relationship between candidate descriptions in cluster $U_1$ with candidate descriptions in cluster $U_2$. Listing 3.6 shows how relationship confidence is calculated for each object property.

```
#From cluster $U_1$
RelConf(foaf:made) = \frac{RelInst(foaf:made)}{3} = \frac{3}{3}

#From cluster $U_2$
RelConf(mo:maker) = \frac{RelInst(mo:maker)}{3} = \frac{3}{3}

RelConf(mo:track) = \frac{RelInst(mo:track)}{3} = \frac{3}{3}
```

Listing 3.6: Annotate clusters with evidence for object properties.

Note that object properties may relate a URI resource with anonymous resources expressed as blank nodes, local to the given RDF graph. In such cases, our methodology searches for the following: (i) evidence of a multi-valued attribute, (ii) evidence of a composite attribute, and (iii) evidence of a relationship with another class.
3.3.5 Infer a Conceptual Structure

To infer a conceptual description for an RDF graph, our methodology considers as evidence the annotations derived for the cluster analysis step. As previously discussed, each cluster suggests a class using a simple heuristic. For each cluster, our approach selects the class with the highest confidence value. Table 3.1 shows an example of the kinds of annotations discovered for each cluster to support the inference of a conceptual description. Using the ER model constructs, classes inferred from each cluster are represented as entity types:

<table>
<thead>
<tr>
<th>cluster</th>
<th>class</th>
<th>ClassConf</th>
</tr>
</thead>
<tbody>
<tr>
<td>( U_1 )</td>
<td>mo:MusicArtist</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>foaf:Person</td>
<td>0.33</td>
</tr>
<tr>
<td>( U_2 )</td>
<td>mo:Record</td>
<td>0.66</td>
</tr>
</tbody>
</table>

Table 3.1: Suggested classes from a cluster’s metadata annotations.

To infer the kinds of properties that describe an inferred concept, our methodology consults the evidence collected for each cluster. We distinguish between the following types of attributes, and represent them using ER model constructs: (i) required attributes, i.e., attributes that must contain a value for each instance; (ii) multi-valued attributes, i.e., attributes that may contain multiple values; (iii) composite attributes, i.e., attributes that contain meaningful component parts; and (iv) optional attributes, i.e., attributes that do not require a value for each instance. As an example, consider Table 3.2, which shows the kinds of attributes suggested from each cluster along with their respective confidence value.

Properties that have the maximum confidence value (i.e., 1) are considered to be required attributes, otherwise they are considered optional. Blank nodes are likely to suggest composite attributes, whereas multi-valued attributes are suggested by the predicate usage set (see Definition 14) of a candidate description, when the count part of a \( \langle \text{term}, \text{count} \rangle \) pair is greater that one.

As regards the inference of relationships between inferred concepts, our approach consults the object properties discovered for each cluster, and selects those with the highest confidence value. For example, Table 3.3 shows that some candidate descriptions from cluster \( U_1 \) suggest the existence of a relationship between concept mo:MusicArtist and an Unknown cluster.
### Table 3.2: Suggested properties from a cluster’s metadata annotations.

<table>
<thead>
<tr>
<th>cluster</th>
<th>property</th>
<th>mutli-valued?</th>
<th>composite?</th>
<th>optional?</th>
<th>PropConf</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>foaf:name</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>foaf:homepage</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>mo:biography</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$U_2$</td>
<td>mo:image</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>dc:title</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>dc:date</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Table 3.3: Suggested relationships from a cluster’s metadata annotations.

<table>
<thead>
<tr>
<th>cluster</th>
<th>obj. prop.</th>
<th>domain</th>
<th>range</th>
<th>suggested_by</th>
<th>RelConf</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>mo:made</td>
<td>$U_1$ (mo:MusicArtist)</td>
<td>$U_2$ (mo:Record)</td>
<td>${cd_1, cd_2, cd_3, \ldots}$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>mo:made</td>
<td>$U_1$ (mo:MusicArtist)</td>
<td>$U_i$ (Unknown)</td>
<td>${cd_i, \ldots}$</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$U_2$</td>
<td>mo:maker</td>
<td>$U_2$ (mo:Record)</td>
<td>$U_1$ (mo:MusicArtist)</td>
<td>${cd_4, cd_5, cd_6, \ldots}$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>mo:track</td>
<td>$U_2$ (mo:Record)</td>
<td>$U_i$ (mo:Track)</td>
<td>${cd_4, cd_5, cd_6, \ldots}$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>mo:available_as</td>
<td>$U_2$ (mo:Record)</td>
<td>$U_i$ (mo:Torrent)</td>
<td>${cd_4, cd_5, cd_6, \ldots}$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Note that the confidence value suggests the proportion of candidate descriptions that have verified the existence of a relationship between the concepts derived. In such cases, our approach infers the relationship with the highest confidence value.

More detailed examples of inferred schemas expressed with the ER model are shown in Figure 2.3. The next section describes an empirical evaluation of the implemented schema inference methodology, using different scenarios.

### 3.4 Experimental Evaluation

This section presents an empirical evaluation of the schema inference methodology. For the purposes of the evaluation, a number of different LD datasets were selected based on their publication methodologies. The evaluation distinguishes between two types of LD datasets: (i) those generated from mapping relational databases to RDF using a systematic approach, such as the D2RServer tool [Bizer and Cyganiak, 2006]; and (ii) published LD datasets from the WoD. Table 3.4 shows a categorisation of the LD datasets that participated in the evaluation studies, grouped by publication methodology:

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>of triples</th>
<th>of classes</th>
<th>BNodes?</th>
<th>published by</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CDShop</td>
<td>303</td>
<td>3</td>
<td>Y</td>
<td>D2RServer</td>
</tr>
<tr>
<td>2</td>
<td>Conference</td>
<td>300</td>
<td>8</td>
<td>Y</td>
<td>D2RServer</td>
</tr>
<tr>
<td>3</td>
<td>Birt_DB⁴</td>
<td>28.5K</td>
<td>9</td>
<td>N</td>
<td>D2RServer</td>
</tr>
<tr>
<td>4</td>
<td>Jamendo⁵</td>
<td>1.1M</td>
<td>11</td>
<td>N</td>
<td>DBTune.org</td>
</tr>
<tr>
<td>5</td>
<td>Magnatune⁶</td>
<td>322K</td>
<td>7</td>
<td>N</td>
<td>DBTune.org</td>
</tr>
</tbody>
</table>

Table 3.4: Linked data sources participating in evaluation.

As previously discussed, it is important for the schema inference techniques to obtain the best possible partitioning of the pool of individuals. This is influenced by the choice of the linkage scheme used to characterise the similarity of a pair of clusters, as well as the final clustering determined by the overall silhouette coefficient. For such reasons, the evaluation methodology presented has a two-fold purpose: (i) to study whether individuals have been assigned to the actual clusters according to some gold standard (i.e., measuring the quality for a given

⁴http://eclipse.org/birt/phoenix/db; retr. 2013/06/03
⁵http://dbtune.org/jamendo; retr. 2014/01/04
⁶http://dbtune.org/magnatune; retr. 2014/01/04
clustering result); and (ii) to determine the extent to which the schema inference methodology can infer a conceptual summary by identifying the correct classes, properties and relationships.

**Methodology and Metrics**

To measure the quality of the final clustering the *F Score measure* was used [Larsen and Aone, 1999]. A prerequisite of this measure is knowledge of which individual belongs to which cluster identified by a class label e.g., MusicArtist. We asked human reviewers\(^6\) to undertake the task of manually assigning a list of RDF instances (represented as candidate descriptions) to each cluster, providing them with an appropriate list of class labels. This process results in a gold standard. The next step was to measure the quality of the outcome using our methodology. We then compared the candidate descriptions generated by the algorithm with the gold standard. For each particular class label \(L_r\) of size \(n_r\) and cluster \(U_i\) of size \(n_i\), with \(n_{ri}\) denoting the number of individuals in cluster \(U_i\) that belongs to \(L_r\), the *F Score measure* of this class and cluster is defined as,

\[
\text{F Score}(L_r, U_i) = \frac{2 \times P(L_r, U_i) \times R(L_r, U_i)}{P(L_r, U_i) + R(L_r, U_i)} \in [0, 1].
\]

In the FScore measure, \(P(L_r, U_i)\) is the precision value, computed as \(n_{ri}/n_i\) and \(R(L_r, U_i)\) is the recall value, computed as \(n_{ri}/n_r\), for the class \(L_r\) and cluster \(U_i\). The FScore of the class \(L_r\) is the maximum FScore computed.

The goal is to measure how accurate the clustering result is. An ideal clustering solution will be the one in which every class inferred by our algorithm has a corresponding cluster, with exactly the same elements, as in the gold standard. In such a case, the FScore measure will be at its maximum value, i.e., 1. In general, the higher the value of the FScore, the better the quality of the derived clustering.

**Quality of Inferred Schemas.** For the second part of the empirical evaluation, we used common quality measures from the information retrieval literature, viz., Precision, Recall and F-measure [Baeza-Yates and Ribeiro-Neto, 1999]. The purpose of this part of the experimental studies was to evaluate the quality of

---

\(^6\)For this experiment we asked four reviewers all experts on linked data and data integration tasks.
an inferred schema. More specifically, the ER constructs (i.e., entity types, attributes and relationships) inferred by the developed methodology are compared with the expected schemas that were designed manually by the same human reviewers as from the previous experiment. To observe the structure and to obtain a view of how data are organised in each of the datasets from Table 3.4, several exploratory queries were used to assist human reviewers to understand how concepts in a data source are described and the relationships among them. The common ER model has been used to represent the schemas in terms of its constructs i.e., entity types, attributes and binary relationships. In cases where data sources have been published from a direct mapping from relational databases, and in cases that we had access to, the actual SQL schemas provided sufficient evidence to describe the schemas with the ER model.

For each schema derived, and each of the ER model constructs i.e., entity types (ET), attributes (AT), and binary relationships (R), the experiment measures: precision, as the proportion of relevant constructs extracted among extracted constructs; that is:

\[
\text{Precision} = \frac{|TP|}{|TP| + |FP|}, \tag{3.15}
\]

recall, as the proportion of relevant constructs extracted among all relevant constructs; that is:

\[
\text{Recall} = \frac{|TP|}{|TP| + |FN|}, \text{ and,} \tag{3.16}
\]

F-measure, which combines precision and recall; that is:

\[
\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3.17}
\]

For entity types, we determine true positives (TP), i.e., entity types needed and inferred, false positives (FP), i.e., entity types not needed but inferred, and false negatives (FN), i.e., entity types needed but not inferred. For attributes and special kinds of attributes (e.g., composite, multi-valued) we determine TP, i.e., attributes needed and inferred as attributes belonging to the correct entity type, FP, i.e., not needed for an entity type but inferred, and FN, i.e., any missing attributes. Finally, for binary relationships we determine TP, i.e., relationships inferred involving the correct entity types, FP, i.e., relationships incorrectly inferred between entity types, and FN, i.e., any missing relationships.
Determining Clustering Parameters

Section 3.1 outlined the steps that comprise a typical hierarchical clustering algorithm, as well as the different implementations that are influenced by the choices of: the stopping criteria, and the way that the similarity between a pair of clusters is determined. In Section 3.3.3, we introduced three well-known linkage-schemes from the literature. This section analyses the implementation choices for the above parameters. More specifically, Section 3.3.3 described how our hierarchical clustering algorithm terminates when the minimum similarity value for which a pair of clusters can be considered candidates for merging is below a certain threshold.

Before elaborating on the empirical results of the evaluation described for our schema inference methodology, this section firstly discusses the choice of the minimum similarity value for merging a pair of clusters, and finally the choice of the linkage-scheme for determining the similarity between a pair of clusters (i.e., single, complete, and average linkage). For all the experiments subsequently described, the configuration of the developed methodology was based on values that were empirically derived as described below.

Min. threshold for merging. To characterise that a pair of clusters is sufficiently similar to be considered as a candidate for the merging step, a threshold \( t \) must be given as a parameter. We observed the effect of different choices of \( t \) on the maximum average silhouette width and thus on the quality of the clustering. The clustering algorithm was run with different values of \( t \), iteratively increasing its value from \( t = 0 \) until \( t = 1 \), in steps of 0.1. The effects are shown in Figure 3.5, where the x-axis shows different values for \( t \), and the y-axis shows the maximum \( ASW_{overall} \) overall for the LD datasets 1, 2 and 3 from Table 3.4.

As the value of threshold \( t \) moves closer to 1.0, the merging step of the clustering algorithm becomes stricter as regards the choice of which clusters are sufficiently similar to be merged. This means that RDF instances (i.e., individuals) that are not sufficiently similar, always according to the similarity measure used, remain un-clustered. Regarding the value of maximum \( ASW_{overall} \), this means that it will become significantly smaller, since most of the individuals remain in singleton clusters, where the silhouette value for singleton clusters is zero.

Figure 3.5 illustrates the drop of the \( ASW_{overall} \) in cases where the value of \( t \) is around 0.4 and 0.5 respectively for RDF data sources 2 and 3. In cases where
the value for \( t \) moves closer to 0, the maximum \( ASW_{overall} \) seems to remain constant. This is perhaps because the algorithm is more flexible as to which clusters are sufficiently similar as candidates to be merged, and thus ends up merging fewer similar clusters. When the value of \( t \) is closer to zero, this allows more merging steps to occur, and therefore more iterations before the clustering algorithm terminates. For each merge step, the algorithm calculates the maximum \( ASW_{overall} \) for the resulting clustering. The results of this experiment suggest that varying \( t \) does not always influence the choice of the maximum \( ASW_{overall} \).

For example, this is shown in the CDShop dataset from Figure 3.5 (i.e., RDF data source 1). In this particular dataset, it seems that there is a clear distinction between the number of properties used and the predicates used to describe each resource which is an instantiation of each class. This enables the similarity measure to distinguish which of the RDF instances are highly similar. For example, instances of the class \texttt{eShop:CD} are always described with the following set, comprised of 6 predicates: \{\texttt{eShop:interpret, eShop:album, eShop:price, eShop:cover, eShop:category, eShop:track}\}. There are no RDF instances that are missing any predicates or using any predicates that are also used to describe instances of some other class, and all the RDF instances are typed using the \texttt{rdf:type} predicate.
This allows one to conclude that, in cases where most or all of the individuals from each class are using the same or almost the same set of predicates, the choice of \( t \) does not influence the value of \( \text{ASW}_{\text{overall}} \). The algorithm is more certain with regards to the number of clusters, and thus the partitioning of the data. However, since we cannot predict the degree of conformance the data described in an RDF data source possesses, it seems that a very strict value for \( t \) (i.e., closer to 1) is not suitable; this would cause more individuals to remain in their singleton clusters, at the same time significantly reducing the value of the \( \text{ASW}_{\text{overall}} \). Therefore, from this empirical experiment, it seems reasonable to set the value for \( t \) to be in the range of \( t \in [0.4, 0.5] \). In the experiments to follow, the value of \( t \) was set to \( t = 0.45 \), which justified empirically to be a reasonable choice.

**Linkage schemes.** In addition to obtaining an empirical value for the termination condition, we empirically observed the behaviour of our clustering algorithm using the different linkage schemes (as described in Section 3.3.3). Thus, for the RDF data sources 1, 2 and 3 (from Table 3.4), we observed how the quality of the clustering (measured with the \( \text{ASW}_{\text{overall}} \)) and thus the suggested number of clusters, is influenced by the choice of the linkage scheme. As illustrated in Figure 3.6, it seems that in most cases the different linkage schemes concur in identifying the appropriate number of clusters, with the exception of complete linkage. For example, Figure 3.6(b) shows that, using complete linkage, the maximum \( \text{ASW}_{\text{overall}} \) occurs when the number of clusters is 14, when in fact the real number of clusters in the dataset is 9. Conversely, alternative linkage schemes identified 12 clusters as the real number of clusters in the data, which is closer to the real number of clusters.

For our purposes it seems reasonable to choose the single or average linkage schemes which, as shown in Figure 3.6, generally have identical or very similar results. In the experimental evaluation to follow, the average linkage was used as the default linkage scheme for the clustering algorithm when identifying the similarity between a pair of clusters.

### 3.4.1 Experiment 1: Reverse Engineering

During the early days of bootstrapping the WoD, relational databases played a key role in expanding the WoD as a source of data to be published according to the LD principles. Since then, several tools, such as the D2RServer [Bizer and Cyganiak,
Figure 3.6: Different linkage schemes, the effect on the top-k maximum average silhouette width and the suggested number of clusters.
2006], and mapping languages, such as described in [Arenas et al., 2012; Das et al., 2012], have been used in the translation of data structured under relational tables into RDF graphs, published in the form of RDF triples using standardised techniques. However, such datasets lack an explicitly defined schema, which previously existed in the relational database. The purpose of this experimental scenario is to study the effectiveness of our schema inference methodology to reverse engineer the structure of RDF graphs that were automatically generated from relational databases. RDF data sources 1, 2 and 3 in Table 3.4 participate in this study.

As described at the beginning of Section 3.4, our experimental methodology has a two-fold purpose. Therefore, before determining the extent to which the schema inference methodology is able to derive a schema from an RDF graph, we begin by considering how well our clustering technique is able to identify the correct clusters. This is crucial since the schema inference methodology aims to reveal common structural patterns of the data from the clusters formed during the process of inferring a structural summary over explicitly-defined RDF instances.

We measured the quality of the clustering using the FScore measure (see Equation 3.14). Figure 3.7 shows the results using the CDShop (RDF data source 1 from Table 3.4). According to the gold standard, the data from this dataset is partitioned into 3 clusters. The clustering algorithm with the use of the silhouette coefficient has successfully identified all individuals of the class labels Track, and the CD into the correct groups, except those from the Category label. The reason behind this is the fact that instances of the Category class are described as blank nodes without any RDF typing information. According to Section 3.3.4, the schema inference methodology interprets blank nodes without RDF typing information as composite attributes, using the ER model constructs. Although individuals of the Category class are not assigned to a separate cluster, the evidence of a composite attribute allows SPARQL queries to be formulated that can populate data from the Category class (for an example, see Listing 3.7).

```
SELECT DISTINCT ?o ?category
WHERE {
    eShop:CdNo9 eShop:category ?c .
    ?c eShop:name ?category.
}
```

Listing 3.7: Exploring triples of the Category class.
Figure 3.7: Quality of resulted clustering using CDShop data source.

Figure 3.8: Quality of resulted clustering using Conference data source.

Figure 3.9: Quality of resulted clustering using Birt_DB data source.
A similar effect was observed with the data published under the *Conference* dataset (Figure 3.8). The individuals from the *Topics* class are described locally in the RDF graph as blank nodes without any RDF typing information, and thus are interpreted as a composite attribute. Another observation is that, for some class labels (shown in the x-axis), the values of the FScore drops due to the fact that several individuals are described with the same or almost the same predicates; they are, however, typed as instances of different classes using `rdf:type` RDF statements. Consequently, individuals that are not instantiations of the same class are assigned to the wrong cluster (causing a drop in recall) or in the same cluster (causing a drop in precision). As an example of this, individuals of both *Researcher* and *PhDStudent* share the predicates `{iswc:first_Name, iswc:last_Name, iswc:address, iswc:homepage}`. The clustering algorithm assigned some individuals of *Researcher* with the class label of *PhDStudent*, and vice versa. In addition, there are cases where individuals of both *Researcher* and *PhDStudent* co-exist in the same cluster.

Moreover, in this scenario we observed the case where an RDF data source has no blank nodes, however, in some cases several predicates are used to describe individuals of a different class. Figure 3.9 shows the results from measuring the clustering quality from the Birt_DB (RDF data source 3 from Table 3.4). In this case, although all the individuals assigned to each class label and cluster should belong to that cluster, thereby maintaining the precision at its highest value, there are several individuals that should belong to one class label yet are assigned to different clusters, thus being annotated with a different class label (causing a drop in recall). In this case we also noted that several individuals were not assigned to any clusters, causing further drops in recall. The maximum $\text{ASW}\_\text{overall}$ derived for this dataset is responsible for such drops, causing some individuals to remain in their singleton clusters, or the same individuals to be partitioned into different clusters. More specifically, the maximum $\text{ASW}\_\text{overall}$ computed for this dataset is 0.976, suggesting 17 clusters, whereas the second best $\text{ASW}\_\text{overall}$ was 0.960, suggesting 9 clusters, which is in fact the real number of classes that exist in the data.

Overall, the schema inference methodology has been provided with sufficient knowledge to draw conclusions about how the data are structured in the RDF graphs. The values of the silhouette coefficient suggesting the partitioning of the data are reasonable, and do not influence post-processing tasks downstream.
Thus, the schema inference methodology is able to determine good results for the inferred structural summaries (see Table 3.5):

<table>
<thead>
<tr>
<th></th>
<th>CDShop</th>
<th></th>
<th>Conference</th>
<th></th>
<th>Birt_DB</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ET</td>
<td>AT</td>
<td>R</td>
<td>ET</td>
<td>AT</td>
<td>R</td>
</tr>
<tr>
<td>Precision</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.90</td>
<td>0.92</td>
</tr>
<tr>
<td>Recall</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.90</td>
<td>0.92</td>
<td>0.92</td>
</tr>
<tr>
<td>F-measure</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.95</td>
<td>0.92</td>
<td>0.92</td>
</tr>
</tbody>
</table>

Table 3.5: Quality of inferred schema, where (ET): Entity Types, (AT): Attributes, and (R): Relationships.

To conclude this experimental scenario, we now observe the quality of the inferred structural summaries in terms of identifying the correct classes, their properties and relationships. For this study, the measures described in Equations 3.15, 3.16 and 3.17 were used. The results are presented in Table 3.5, where each of the derived schema constructs represented with the ER model were compared with the gold standard. We observed that the schema inference methodology managed to infer a structure as expected, with minor flaws. More specifically, for all the RDF data sources participating in this experimental scenario, the methodology managed to correctly infer all the entity types (i.e., classes) expected. For some classes that had individuals described as blank nodes, the inferred schema created a special type of attribute instead (i.e., a composite attribute), therefore, we do not classify missing entity types as false negatives.

As discussed previously in Section 3.3.3, missing attributes create complications for the schema inference process. In this scenario, data from relational tables were mapped to RDF triples; instances that had NULL values for some attributes are missing from the descriptions of the RDF instances. This effect slightly lowers the performance of the methodology in identifying the correct attributes that describe each entity type. As regards the identification of binary relationships, the algorithm performs well; however, is worth noting that some false positives were found in cases where there is evidence that classes occur in a class hierarchy (i.e., subsumption relationships). As we pointed out earlier, extending our methodology to encompass class hierarchies has been left for future work.

RDF data sources are diverse in terms of representing the data with different structures and terminologies from various LD vocabularies; nevertheless, the
schema inference methodology as of the empirical evaluation just discussed performed well in cases where the RDF graphs have been translated from relational databases. The evaluation scenario that follows seek to study empirically the effectiveness of the methodology on RDF graphs that have been published using publishing methodologies that did not assume the existence of an explicit schema.

### 3.4.2 Experiment 2: Sources from the Web of Data

The purpose of this experimental scenario was to measure the extent to which the schema inference methodology can identify a schema of good quality by considering real LD datasets from the WoD that were not generated to conform to the structure of a previously-defined schema. For this experiment, two LD datasets published under the DBTune.org project, viz., Jamendo and Magnatune, were chosen to participate in this study.

As in the previous experimental scenario, we measure the quality of the inferred schema using precision, recall and F-measure with Equations 3.15, 3.16 and 3.17 respectively. Figure 3.10 shows the results from inferring a structural summary for the Jamendo dataset. For this dataset, we noticed that several RDF instances are not annotated with any typing information using `rdf:type` statements. Without any typing knowledge, such individuals are grouped in their appropriate cluster based on similarities. However, from the explicit RDF statements, nothing can be said regarding class memberships. In such cases, the developed methodology assigns untyped individuals to several specialised clusters with an `unknown` label.

As regards the ER model used to describe our inferred schemas, the entity types inferred from such clusters are assigned to the `unknown` label. This, however, does not restrict the methodology for inferring the structure of an entity type. The methodology is able to infer the attributes that describe such entity types, as well as whether such entity types participate in any relationships with other entity types. Such structural knowledge enables the formulation of SPARQL queries that can retrieve data from RDF instances that have no typing information. Having inferred the structure of entity types that are labelled as `unknown`, it might be possible to implicitly obtain the actual typing information by using the inference semantics of RDFS and OWL that were used to define the terms used to describe the data in an RDF graph. This improvement was left for future work.
Figure 3.10: Quality of inferred schema for *Jamendo*.

Figure 3.11: Quality of inferred schema for *Magnatune*.
Although entity types with the unknown label are discovered by the approach, the evaluation results shown in Figure 3.10 consider such entity types as false positives. This causes a slight decrease in the precision of the entity types inferred. Similarly, any binary relationships inferred for unknown entity types are considered as false positives, because the technique is unaware of which entity types are participating in the relationship. Thus, there is a slight decrease in the precision for relationships. In addition, in this dataset some relationships were missing from the results e.g., the relationship event:factor between the mo:Lyrics and the mo:Performance entity types. Missing relationships are considered as false negatives, hence the slight decrease in the recall measure for relationships.

As another example, this scenario considered the Magnatune dataset; the results from evaluating the quality of the inferred schema for this LD dataset are reported in Figure 3.11. According to the gold standard for this dataset, the schema inference methodology was able to infer the expected structural constructs, and thus achieved maximum precision and recall. Overall, from our empirical methodology described in this section, we observed that RDF data sources use a small number of classes (i.e., rdfs:Class or owl:Class) to describe their resources, and that the schema inference methodology described in this dissertation performs well by algorithmically inferring structural summaries of good quality, which can support the automatic integration of RDF data sources.

### 3.5 Related Work

As previously discussed, the WoD can be a challenging environment for consumers seeking a coherent and integrated view over a multitude of LD datasets. Several challenges contribute to making the integration of LD datasets a difficult task, some of which are: (i) the challenge of locating which LD datasets can contribute answers to a user query; (ii) the lack of conceptual description of LD datasets; and (iii) dealing with scalability issues.

These challenges can be addressed more easily if conceptual summaries of RDF data sources can be algorithmically derived. This section begins by discussing how work presented in this chapter relates to existing work on locating LD datasets that could contribute answers to a given query. It then moves on to discuss related work on discovering RDF-specific structural knowledge from LD datasets, and finally discusses how proposals for distributed query processing could benefit
from the existence of conceptual descriptions. The developed methodology for inferring structural summaries is positioned in relation to other proposals for discovering structural knowledge of datasets, which can inform the formulation of SPARQL queries over RDF data sources.

3.5.1 Source Discovery

There exists ongoing research on providing summaries or descriptions of what kind of data can be found in which RDF data sources using index structures. Konrath et al. [2012] describe SchemEX, an enhanced index structure that leverages RDF typing information and links to extract schema-level knowledge from a stream of RDF triples that are traversed from an RDF graph using a fixed-window. Schema knowledge from the RDF graph is then used for the construction of schema-level indices that describe the structure of the data in an LD dataset. Schematic information from several datasets is linked with the index structure. Given a SPARQL query, the index structure allows SchemEX to locate which datasets contain instances of a specific RDF class, which could potentially contribute data to answer the query. In this respect, SchemEX deals with the challenge by providing a summary of which triples can be located in which dataset.

By contrast, the schema inference methodology presented in this chapter does not utilise the discovered schema knowledge to build an index structure, which is then used to guide query processing by locating relevant LD dataset that can contribute to answering a given query. Instead, our methodology builds on cluster analysis, so as to reveal common structural patterns by inspecting RDF instances from each cluster, which will give rise to the structure of concepts that exist in single RDF sources as well as internal relationships with other concepts. Our approach, however, does not dereference any links to discover more structures from linked LD datasets. Similarly to SchemEX, however, our methodology leverages RDF typing information to infer class labels for discovered concepts with the aim of supporting query execution. As regards evaluation, SchemEX presents evidence of a good trade-off between scalability and result quality. Although a direct comparison is not applicable to our structure inference methodology, we recognise that the clustering approach, on which our techniques are built, might not scale well for very large RDF graphs.

SchemEX constructs a schema-level index, whereas Harth et al. [2010] propose a methodology for constructing instance level indices, using an approximate
multidimensional indexing structure (i.e., QTrees). The construction of the index structure involves the use of a hash function over individual components of RDF triples that exist in LD datasets, in order to obtain data points that correspond to a three-dimensional QTree. To identify similar RDF triples, minimal bounding boxes (MMBs) are used to approximate sets of similar triples. Given a SPARQL query, a set of MBBs is returned for each triple pattern in the query which suggests relevant LD datasets that can contribute answers to the query result. Our methodology for inferring structural summaries is somewhat related insofar as it uses clustering to group together similar RDF instances, which is somewhat similar to the MMBs approximated in a three-dimensional space. However, our methodology differs in that it has an understanding of how concepts are represented in an RDF source, as well as an understanding of which datasets organise which triples. Thus, the emphasis in [Harth et al., 2010], as exploited in [Prasser et al., 2012], is on providing a summary of RDF triples at the instance level that can inform efficient query evaluation, whereas the emphasis on the work presented in this chapter is on providing a schema-level summary that can inform data integration.

### 3.5.2 Knowledge Discovery and Ontology Mining

One approach used in the literature for extracting schema information from graphs is to use bisimulation, which has been proposed for schema extraction from XML data. Briefly, bisimulation is used to describe equivalent nodes based on their neighbourhood [Abiteboul et al., 1999]. [Khatchadourian and Consens, 2010] proposed the use of a bisimulation-based structure for deriving structural summaries for RDF graphs with ExpLOD, an approach for understanding the RDF usage of a given RDF graph. The resultant structural summaries contain structural meta data about the RDF graph that consist of the set of instantiated classes, or the set of predicates used by the resources. The ExpLOD approach builds on the use of text labels and sub-graphs of RDF data created using SPARQL queries. The local-parts of URIs are used to label the bisimulation graphs, whereas in our developed methodology we use a distance function that utilises local names to judge whether a pair of RDF instances are similar. In addition, our schema inference methodology makes use of the class usage of each RDF instance to infer entity types for each cluster. Moreover, and in line with our approach, ExpLOD makes use of RDF typing information.
Paulheim and Bizer [2013] proposed \textit{SDType}, a heuristic type inference mechanism that exploits links between instances to suggest their typing information. This approach uses an object classification approach, based on links between resources as indicators for inferring their types. In addition, SDType builds on probability distributions to assign types. Although this approach solves a different problem to ours, we noted that the inference of types for un-typed RDF instances could be used as extra evidence for our methodology for inferring the types our entity types labelled with the \textit{unknown} label.

Zong et al. [2012] explored a methodology to dynamically construct a concept hierarchy using LD datasets from the bio-medical domain. A similar methodology to ours is used, based on a hierarchical clustering approach to group similar instance level data. The suggested distance function computes pairwise similarities based on predicate values that are RDF links, whereas our similarity measure utilises the local names from literal triples or RDF links combined with knowledge from the class usage sets. Similarly to our technique, RDF typing information is used to suggest relevant concept names. The approach assumes that the LD dataset is described using a single ontology whereas our techniques does not make such assumptions. With respect to our schema inference methodology, RDF instances grouped into clusters are utilised to reveal common structural patterns. In addition, their described technique is restricted in terms of dynamically inferring relationships between the inferred concepts, whereas in our proposal the discovery of domain/range axioms is made possible: such information is captured as binary entity type relationships.

Another piece of related work is the proposal described in [Völker and Niepert, 2011] on mining ontologies from RDF instance level data, described as \textit{Statistical Schema Induction}. In contrast to our methodology, which builds on cluster analysis, statistical schema induction builds on mining association rules from RDF data sources to suggest schema-level knowledge. Association rules that satisfy a user-provided confidence threshold contribute to the construction of simple ontologies suggesting the concepts and predicates used. In our methodology, instead of mining association rules, silhouette coefficients are used to suggest a partitioning of the space into groups of related individuals, which are then explored to infer a structural summary. For the statistical schema induction, resources at the instance level are required to have RDF typing information. This is equally useful for our schema inference methodology; however, the lack of RDF typing
information does not restrict our approach, since the resources are still organised into special kinds of clusters which can guide the development of a conceptual description and reveal the structure of the data.

### 3.5.3 Distributed Query Processing

As previously outlined, conceptual descriptions can inform data integration. Similarly, having an understanding of the conceptual structure of RDF data sources can also support distributed query processing over RDF data sources. Distributed query processing requires an understanding of how concepts are structured in an RDF source; however, such knowledge is typically not available for RDF data sources. Quilitz and Leser [2008] describe a federated query engine called DARQ which employs handcrafted source descriptions, so-called service descriptions. Service descriptions contain information about the RDF predicates that are used to describe the data in an RDF source, which can also include additional statistical information.

Other federated approaches [Görlitz and Staab, 2011] use voiD descriptions, which can be considered a schema, depending on their level of detail. However, voiD descriptions focus more on describing statistics about an RDF data source, and do not provide complete descriptions of how concepts are structured. In contrast, the inferred structural summaries derived by our schema inference methodology can be used to assist distributed query processing, providing conceptual summaries that can be inferred on the fly.

Approaches such as FedX [Schwarte et al., 2011b] do not require any meta data upfront; instead, FedX relies on SPARQL ASK queries for sources selection at query time to annotate triple patterns in the query with relevant sources. The schema inference approach described can be used to suggest conceptual descriptions that can inform SPARQL query formulation.

### 3.6 Discussion and Conclusions

This chapter has discussed how a hierarchical clustering algorithm may be employed to automatically infer a conceptual structure to be used for the integration of multiple schema-less RDF data sources. In terms of implementation, the result of the INFERSCHEMA operator is a set of schema constructs that are mapped to the generic constructs of the DSToolkit’s canonical model. The ER model used
here, allows a rich description of the inferred schemas along with useful meta
data (such as the knowledge of which attributes are multi-valued or composite) to support downstream tasks such as the identification of matches, the derivation of mappings, and to inform SPARQL querying over the underlying RDF data sources.

In addition, the inferred knowledge from the clusters can potentially be used to guide the automatic construction of simple LD ontologies over specific RDF graphs. For example, entity types can suggest the definition of URI resources as *classes* which are typed using meta-classes such as `rdfs:Class` or `owl:Class`. Moreover, the different kinds of attributes inferred can potentially inform the definition of resources as properties. For example, data properties can be defined with `rdf:Property` or `owl:DatatypeProperty` meta-properties, and binary relationships between concepts can inform object properties defined with `owl:ObjectProperty`, where the domain and range can be declared via RDFS meta-properties i.e., `rdfs:domain` and `rdfs:range` respectively.

Although the experimental evaluation designed for evaluating the methodology performed well in cases where sources had sufficient RDF typing information and were of manageable sizes, several improvements to the techniques developed were highlighted for future work.

On a different note, recent research by [Schmidt and Lausen, 2013] discussed the fact that the LD principles suggest a minimalistic way of publishing and interlinking data; however, for many applications, guarantees on the structure of an RDF data source may be essential. To bridge this gap they proposed RDF Data Description (RDD), a language for imposing data constraints over RDF graphs similarly to DTDs in XML. We noticed that the inferred structural summaries could potentially be used to assist human developers with describing RDF data sources with RDD documents, which aims to bridge the schema-less gap of RDF data sources in a similar way to our work.

Although the knowledge from conceptual descriptions could perhaps be applicable in different contexts, in this dissertation we propose that such descriptions can inform data integration tasks over multiple RDF data sources. In a dataspace system, knowledge of conceptual descriptions can support downstream tasks such as semantic matching, derivation of mappings and query rooting. More specifically, during the matching stage, schemas are utilised as sources of evidence to inform the derivation of correspondences between schema elements. However,
such decisions are highly uncertain for several reasons, as we shall discuss in
the next chapter. As has emerged from previous chapters, the management of
uncertainty in different stages of a dataspace is desirable to support the ethos
of a self-produced system that builds on automation. The next chapter elabo-
rates on a methodology for reasoning under uncertain situations in relation to
the matching stage of a dataspace.
Chapter 4

Managing uncertainty in LD matching

“The true logic for this world is the calculus of Probabilities, which takes account of the magnitude of the probability which is, or ought to be, in a reasonable man’s mind.”

James Clerk Maxwell, 1850.

Chapter 3 contributed a cluster-based approach to address the schema-less nature of LD. It described a methodology for automatically inferring structural summaries (i.e., an ER schema) of LD datasets. Having a schema to which a RDF dataset adheres is not a requirement either of the RDF standard or of the LD principles. The majority of LD datasets currently published probably lack something one could call a schema. However, knowledge of schemas is useful for the data integration processes. More specifically, a pay-as-you-go data integration process (as envisioned by dataspaces) often relies on automatic derivation of schema mappings that specify the semantic relationships between the source datasets and the mediated schema [Sarma et al., 2011]. This automation is beneficial as it provides integration-on-demand without major costs, but it introduces uncertainty\(^1\). For example, the inferred schema could be uncertain due to less than perfect performance by the clustering technique described in Chapter 3 (e.g., caused by the similarity measure) used to derive the groups of instances from which a best-effort structure of the LD dataset is derived. In addition, the matching stage of the datasource life-cycle also uses automatic techniques to

\(^1\)For a discussion on uncertainty in dataspaces the reader is referred to Section 2.6.2
derive associations between the schemas to be integrated. The decisions made using such matching algorithms are uncertain due to either lack of knowledge or to less than perfect performance of the algorithms used. In fact, matching is, in general, an error-prone phase [Gal, 2006] which is known to cause decisions to be uncertain [Gal and Shvaiko, 2009]. This has an immediate compound effect on the derivation of the schema mappings during the mapping stage of the life-cycle.

Several sources of uncertainty as well as the challenges this raises at different life-cycle phases in pay-as-you-go data integration systems were discussed in Section 2.6.2. Broadly speaking, it is not clear how one should quantify uncertainty (i.e., or, even more basically, how uncertainty should be modelled) nor how one should reason under uncertainty. In this context, this chapter contributes a principled methodology for quantifying uncertainty and for reasoning with uncertain knowledge using Bayesian updating [Pearl, 1988, pp.29–40]. The contributed techniques have been applied to quantifying uncertainty at the matching stage and, in particular, on how to reason about the equivalence of a pair of schema elements given different kinds of evidence, such as the trustworthiness of similarity scores derived by a set of matching algorithms (referred to from now on as matchers). In a dataspace setting, during the bootstrapping phase, different matchers try to exploit different pieces of knowledge at different levels (e.g., schema, instances) to inform the derivation of a correspondence between schema elements. Their decisions are assigned a degree of confidence in the form of a similarity score. We use similarity scores derived by matchers as pieces of evidence.

Our contribution also uses another form of evidence. A dataspace for LD sources may have available semantic knowledge from LD vocabularies in the form of semantic annotations on the data in LD datasets. The fact that LD datasets are described using shared vocabularies (modelled as ontologies) presents an opportunity to bring together evidence at both the syntactic and the semantic levels, i.e., not just the names and tokens, but also the semantic annotations that characterise entities at the conceptual level. Additionally to the techniques proposed for empirically learning the likelihoods and reasoning under uncertainty with Bayesian updating, this chapter shows how semantic annotations from LD vocabularies can have an impact on improving the outcome of uncertain decision making by matchers that typically work on the syntactic level alone.

The contributed techniques construe uncertainty as subjective probability (i.e.,
degrees of belief). In Bayesian updating, as discussed in subsequent sections, subjective beliefs are used to judge a hypothesis under partial knowledge. The more evidence that is acquired in support of the hypothesis, the more likely that the hypothesis converges to truth (correspondingly the hypothesis converges to falsity if the acquired evidence supports negating the hypothesis). In a dataspace life-cycle, knowledge that can influence the hypothesis of construct equivalence may be available at different stages as the dataspace evolves. It is, therefore, critical that evidence is dynamically and incrementally assimilated with the degrees of belief being updated as evidence accumulates.

In more detail, this chapter describes the following contributions: (a) a generic methodology for reasoning under uncertainty that can be adopted throughout the life-cycle of a dataspace; (b) a set of techniques for initialising the approach through the construction of probabilistic models for the similarity scores that are returned by string-based matchers and for semantic annotations from LD vocabularies; (c) a methodology for assimilating different pieces of evidence in judging the hypothesis of construct equivalence; (d) an application of the above techniques to the matching stage of the dataspace life-cycle, and (e) an empirical evaluation of this particular application aiming to understand whether semantic annotations can improve the uncertain decisions derived by string-based matchers alone.

The remainder of this chapter is structured as follows. Sections 4.1 to 4.3 constitute an overview of concepts on probability theory that our contribution build upon. An overview of the developed framework for updating subjective probabilities in the light of different pieces of evidence is given in Section 4.5. Section 4.6 contributes a technique that uses a kernel estimator to empirically estimate the probability distributions for similarity scores returned by string-based matchers, and Section 4.7 contributes a technique for learning the likelihoods of different kinds of semantic annotations from the WoD. An experimental evaluation of the approach is presented and discussed thoroughly in Section 4.8. Section 4.9 discusses related work. Finally, Section 4.10 concludes with an overall discussion on the proposed methodologies.
4.1 Overview of Probability

This chapter uses probability theory to reason about the truth of a proposition in cases where knowledge is uncertain. A simple method for reasoning under uncertainty is to use Bayesian inference to compute posterior probabilities for a specific hypothesis given the available evidence [Diez and Druzdzel, 2003, pp.880–886]. Probabilities in Bayesian inference are interpreted as stating a subjective view in the form of a degree of belief that a proposition is true. The following sections review material mostly on probability theory.

4.1.1 Probability Theory and Axioms

Probability theory is concerned with studying experiments (or processes) whose outcomes are uncertain [Russell and Norvig, 2010, Ch.13; Durrett, 2010, Ch.1]. Such experiments are referred to as random experiments. Each random experiment is associated with a sample space (denoted by \( \Omega \)), which is the set of all possible outcomes associated with the former. The elements of the sample space are called outcomes, or samples points (denoted by \( \omega \)). A subset of the sample space is called an event, \( E \), i.e., \( E \subseteq \Omega \), and contains those outcomes for which the event holds. Let \( \mathcal{F} \) be a family of subsets of \( \Omega \) with the following characteristics:

(i) \( \emptyset \in \mathcal{F} \)

(ii) if \( E \in \mathcal{F} \) then \( E^c \in \mathcal{F} \), where \( E^c = \Omega \setminus E \) is the complement of \( E \) in \( \Omega \)

(iii) if \( E_1, E_2, \ldots \in \mathcal{F} \) then \( \bigcup_{i=1}^{\infty} E_i \in \mathcal{F} \).

A probability measure \( P \) is then defined as a function \( P : \mathcal{F} \to [0,1] \), which assigns non-negative real values (the probabilities) to events. The function \( P \) is such that the following conditions hold:

(i) \( P(\emptyset) = 0, \, P(\Omega) = 1; \)

(ii) if \( E_1, E_2, \ldots \in \mathcal{F} \), and disjoint events (i.e., if \( E_1 \cap E_2 = \emptyset \)), then

\[
P\left(\bigcup_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty} P(E_i).
\]
Useful Probability Rules. Some theorems follow from the probability axioms above (for more details on the probability theorems and formal proofs the reader is referred to [Howson and Urbach, 2006, pp.16–20]). This dissertation is concerned mostly with conditional probabilities and therefore the following rules are formulated for conditional probabilities. Assuming, the events $E_1, E_2, ...$:

**Rule 1.** The *conditional probability* is:

$$P(E_1|E_2) = \frac{P(E_1 \cap E_2)}{P(E_2)}.$$  

**Rule 2.** The *product rule* is:

$$P(E_1 \cap E_2) = P(E_1|E_2)P(E_2).$$  

**Rule 3.** The *sum rule of probability* is:

$$P(E_1|E_2) + P(\neg E_1|E_2) = 1.$$  

**Rule 4.** Given, $n$ mutually exclusive and exhaustive events $A_1, ..., A_n$ whose probabilities sum to unity, then:

$$P(E) = \sum_{i=1}^{n} P(A_i, E) = \sum_{i=1}^{n} P(E|A_i)P(A_i),$$

where $E$ is an arbitrary event and $P(E|A_i)$ is the conditional probability of $E$ assuming $A_i$ [Papoulis and Pillai, 2002, pp.37–38]. This rule is known as the *law of total probability.*
Rule 5. The assumption of conditional independence, states that each event $E_1, E_2, \ldots$ is independent from some event $A$ [Pearl, 1988, pp.36-37]:

$$P(E_1, \ldots, E_n|A) = \prod_{i=1}^{n} P(E_i|A).$$

These are the basic underlying rules used later on for understanding the Bayesian inference technique discussed in Section 4.4. The interested reader is referred to [Durrett, 2010, Ch.1] for more details on probability theory. For the purposes of this dissertation probability measures are denoted by $P$; we refer to them with the term probability.

### 4.2 Random Variables and Probability Distributions

One can define random variables which associate a numerical value to the outcomes of a random experiment. More formally, a random variable is a function $X : \Omega \rightarrow \mathbb{R}$, where the domain of the function is the outcome space, the target is the real numbers and $X$ is the symbol identifying the function.

In an experiment where $X(\omega)$ can only take a finite or countably infinite number of discrete values the random variable is known as a discrete random variable, whereas in case $X(\omega)$ can take an infinite number of possible values in a continuous interval on the real line the random variable is known as a continuous random variable [Papoulis and Pillai, 2002; Durrett, 2010; Russell and Norvig, 2010]. For a given outcome $\omega \in \Omega$, the function $X$ assigns the measurement value $x \in \mathbb{R}$ to that $\omega$, i.e., $X(\omega) = x$. This chapter will use the simplified notation $X = x$ to indicate that a random variable assigns the value $x$ to the outcome $\omega$.

A random variable can be characterised by its probability distribution function, which associates a probability to the possible values a random variable can take. Let $X$ be a discrete random variable that takes any values in $\{x_1, \ldots, x_n\}$ for $n \in \mathbb{Z}$, the probability that $X$ is equal to $x_i$, i.e., $P(X = x_i)$ is given by the probability distribution function

$$P(X = x_i) = p_X(x_i), \quad \text{for } i \in \{1, \ldots, n\}, \quad (4.1)$$
where \( p_X(x_i) \) is known as the *probability mass function (p.m.f.)*. The total probability of the values the discrete variable can take need to adhere to the following properties:

\[
0 < p_X(x_i) \leq 1, \text{ for all } i; \\
\sum_{i=1}^{m} p_X(x_i) = 1
\]  

Let \( X \) now be a *continuous* random variable. In this case, the number of possible values \( X \) can take is infinite. Thus, the probability \( X = x \) will always be zero. As a result, we are generally concerned with probabilities of intervals such as \( P(a \leq X \leq b) \), where \( a \) and \( b \) are constants. In particular, the probability that the value of \( X \) falls within the interval \([a, b]\) is

\[
P(a \leq X \leq b) = \int_{a}^{b} f_X(x) \, dx,
\]

where \( f_X(x) \) is known as the *probability density function (p.d.f.)*. More precisely, the p.d.f. is a curve that satisfies [Devore, 2011, Ch.4]:

\[
\begin{align*}
&f_X(x) \geq 0 \text{ (non-negativity); } \\
&\int_{-\infty}^{+\infty} f_X(x) \, dx = P(-\infty \leq X \leq +\infty) = 1.
\end{align*}
\]

Graphically, the probability \( P(a \leq X \leq b) \) is found as the area under the curve between the points \( a \) and \( b \).

### 4.3 Conditional Probability Distributions

Extending the above we can introduce *conditional probability distributions*, which provide the probabilities of a random variable taking particular values *given* a certain event has already happened. For *discrete* random variables \( X \) and \( Y \), the conditional probability of \( X = x_i \) given a fixed value of \( Y = y \), i.e., \( P(X = x_i | Y = y) \) is calculated as

\[
P(X = x_i | Y = y) = p_{X|Y}(x_i | y), \quad \text{for } i \in \{1, \ldots, n\},
\]

where \( p_{X|Y}(x_i | y) \) is the *conditional* p.m.f. of \( X \). Similarly, for the case of \( X \) and \( Y \) both *continuous* random variables, the conditional probability of \( X \in [a, b] \)
given a fixed a value \( Y = y \), i.e., \( P(a \leq X \leq b|Y = y) \) is calculated as

\[
P(a \leq X \leq b|y) = \int_{a}^{b} f_{X|Y}(x|y) \, dx,
\]

where \( f_{X|Y}(x|y) \) is the conditional p.d.f. of \( X \).

In the special case where \( X \) is a discrete random variable and \( Y \) is a continuous random variable this results in a conditional probability distribution of a mixed type. Mixed type distributions can either lead to an associated p.m.f. or a p.d.f.; this depends upon whether \( X \) is conditioned on \( Y \) or \( Y \) on \( X \). In particular, for the conditional case \( X|Y \), the conditional p.m.f. is given by

\[
p_{X|Y}(x|y) = \lim_{\Delta y \to 0} P(X = x|y - \Delta y \leq Y \leq y + \Delta y).
\]

(4.7)

Alternatively, for the case \( Y|X \) the conditional p.d.f. is given by

\[
P(a \leq Y \leq b|X) = \int_{a}^{b} f_{Y|X}(y|x) \, dy.
\]

(4.8)

In this dissertation, both cases of the mixed distribution are equally used to derive the likelihood probabilities in the Bayesian inference approach, which is described in more detail in subsequent sections.

### 4.4 Bayesian Inference

Probabilistic inference uses posterior (i.e., conditional) probabilities to perform inductive reasoning for some statement (i.e., a statement of mathematical logic which is either true or false), given some observed knowledge, known as evidence [Russell and Norvig, 2010, pp.490–491]. This dissertation is concerned with a method used for probabilistic inference known as Bayesian inference [Pearl, 1988, pp.29–40; Howson and Urbach, 2006, pp.20–21; Bolstad, 2007, pp.12–16] which introduces a subjective Bayesian view of reasoning under insufficient (or uncertain) knowledge. The underlying logic of this formalism is Bayes’s theorem. Given some hypothesis, denoted by \( H \), and some evidence, denoted by \( e \), Bayesian inference allows one to judge the hypothesis using conditional probabilities. Thus, \( P(H|e) \) denotes the degree of belief in \( H \) under the assumption that \( e \) has been
observed for certain [Pearl, 1988, pp.30–31]. Bayes’s theorem is expressed as:

$$P(H|e) = \frac{P(e|H)P(H)}{P(e)}$$

where $P(H)$ is the prior probability, $P(e|H)$ is the likelihood, and $P(H|e)$ is the posterior probability. The prior is the degree of belief in the hypothesis when no evidence remains to be accounted for. The posterior is the degree of belief in the hypothesis after the new available evidence $e$ is observed. The denominator $P(e)$, often referred as the marginal likelihood, is the degree of belief in the evidence itself, and acts as a normalisation constant [Pearl, 1988, pp.32-34].

The evidence, $e$ has an impact on the computation of the posterior probability for a given hypothesis. This is shown in Equation 4.9: if $P(e|H) > P(e)$ then the posterior probability increases.

**Bayesian Updating.** When multiple sources of evidence are available, Bayes’s theorem can be applied to update or revise a degree of belief derived by the assimilation of evidence in the form of a sequence of observations whenever new evidence is obtained. Bayesian updating can be used to update that prior degree of belief into a posterior, given the new observations. Let $E = \{e_1, e_2, \ldots, e_n\}$ denote a sequence of observations that have already been assimilated. We denote by $\overrightarrow{e_n}$ the sequence $E$ whose last element is $e_n$ and by $\overrightarrow{e_{n-1}}$ the sequence $E \setminus e_n$, i.e., the one whose last element is $e_{n-1}$. If so, the degree of belief is:

$$P(H|\overrightarrow{e_n}) = \frac{P(\overrightarrow{e_n}|H)}{P(\overrightarrow{e_n})} P(H|\overrightarrow{e_{n-1}}).$$

(4.10)

When a new observation $e_{n+1}$ is observed, the objective is to update the prior (i.e., the computed posterior) to account for the new observation. In such a case the new posterior is given by:

$$P(H|\overrightarrow{e_{n+1}}) = \frac{P(\overrightarrow{e_{n+1}}|H)}{P(\overrightarrow{e_{n+1}})} P(H|\overrightarrow{e_n}).$$

(4.11)

Note that the revised posterior $P(H|\overrightarrow{e_{n+1}})$ summarises all the evidence that had been taken into account in the previous posterior $P(H|\overrightarrow{e_n})$ by using it as the new prior. This ability to update the subjective probabilities in the light of new evidence depends on the assumption that the different pieces of evidence are conditionally independent given the hypothesis [Pearl, 1988, pp.37–39].
4.5 Overview of Approach

Bayesian updating allows a previously computed degree of belief for a hypothesis to be adjusted (i.e., increase or decrease) so that it reflects the impact of each piece of evidence as it becomes available. This chapter contributes techniques for assimilating different types of evidence in order to form a degree of belief on a construct-equivalence. These techniques were deployed into the life-cycle of a DSMS (e.g., DSToolkit [Hedeler et al., 2012]) to deal with various aspects of uncertainty in relation to the matching stage. We use the phrase *Bayesian Updating Framework (BUF)* throughout this chapter to refer to the contributed techniques.

Bayesian Updating Framework

Figure 4.1: The assimilation of different pieces of evidence and the update of degree of belief using the *Bayesian Updating Framework*.

Figure 4.1 depicts the BUF. The framework gathers different pieces of evidence in an *evidence pool*. A prerequisite for assimilating evidence is that the likelihoods of each type of evidence are known. The process *empirically derive likelihoods* is responsible for deriving the likelihoods for different kinds of evidence e.g., $P(e|h)$ and $P(e|\neg h)$. Given the likelihoods, Bayes’s theorem requires knowledge of a prior degree of belief. Typically, an initial prior degree of belief can be
derived from a prior distribution function on all the outcomes of the hypothesis. The initial prior simply states our ignorance (i.e., uncertainty) regarding the hypothesis before any evidence is taken into account. In Bayesian inference, there are mainly two categories of prior distributions; informative prior distributions and non-informative prior distributions [Bolstad, 2007, pp.56-62]. The problem of formulating a prior distribution so that an estimation of the prior degree of belief can be derived is beyond the scope of this dissertation (the reader is referred to [Kass and Wasserman, 1996] for a discussion on the selection of prior distributions). Instead, the BUF assigns a simple non-informative prior by adopting the principle of indifference [Kass and Wasserman, 1996; Howson and Urbach, 2006, Ch.9].

The adoption of this principle seems a reasonable choice where there is complete ignorance of prior knowledge regarding a hypothetical judgement, since it assigns a uniform probability to all possible outcomes of the hypothesis. This means that if a given hypothesis has \( N \) possible outcomes, the prior probability that one of the outcomes is observed is \( 1/N \). Given both the prior degree of belief and the likelihoods of each type of evidence, Bayesian updating is used to derive a posterior probability that is incrementally and progressively adjusted as new pieces of evidence become available.

**BUF for Schema Matching in Dataspaces**

To demonstrate how our BUF can be used to dynamically reason on a given hypothesis, we instantiate it in the context of improving decisions in the matching stage of the bootstrapping phase of the dataspace life-cycle. More specifically, this chapter contributes a methodology for managing uncertain decisions arising from the goal of postulating one-to-one semantic correspondences of construct equivalence. Section 4.8 presents the results of evaluating the decisions made by our approach when reasoning over the hypothesis of construct equivalence in the light of different pieces of evidence in relation to matching. The decisions derived by the methodology are compared against the rational decisions made by human experts presented with the same pieces of evidence and asked to judge whether pairs of constructs are equivalent.

As discussed in Chapter 2, the fact that LD datasets are described using shared ontologies presents an opportunity to utilise semantic annotations to improve the decision making of matching processes that, typically, rely on syntactic evidence
The conceptual level annotations (i.e., terminological knowledge) used to make explicit the semantics of entities (i.e., classes or properties) that describe the instance level data in a LD dataset, contain syntactic and semantic knowledge that can be used as evidence for managing uncertainty arising from the need to postulate construct equivalence.

**Definition 1 (Semantic correspondence of equivalence).** Let $S$ and $T$ be conceptual descriptions (i.e., the set of classes or properties that define the structure of the instance level in a LD dataset) of a source and a target LD dataset, respectively. A semantic correspondence of equivalence between a pair of constructs is a triple $\langle c_S, c_T, P(c_S \equiv c_T | E) \rangle$, where $c_S \in S$ and $c_T \in T$ are constructs (i.e., classes or properties) used to describe the data from the datasets, and $P(c_S \equiv c_T | E)$ is the posterior probability representing the degree of belief in the equivalence ($\equiv$) of the constructs given pieces of evidence $E = \{e_1, \ldots, e_n\}$.

Recall from Section 2.3 that we refer to a class as an RDF term that is typed with `rdfs:Class` [Brickley and Guha, 2004] or with `owl:Class` [Mcguinness and van Harmelen, 2004]. Likewise, we refer to a property as an RDF term which is typed with `rdf:Property` or with subclasses of `rdf:Property`; such as `owl:ObjectProperty`, `owl:DatatypeProperty` etc. We often refer to both classes or properties with the more generic term `constructs`.

In terms of implementation, our framework assumes the existence of a conceptual description over the LD sources to be matched and our contributions described in Chapter 3 provide us with the means to obtain such descriptions. The URIs of classes or properties in such descriptions are dereferenced to obtain access to semantic annotations from LD vocabularies. The semantic annotations (i.e., terminological knowledge at the conceptual level) obtained are leveraged in a domain-agnostic way (via the construction of their probability distributions) to reason over the hypothesis of construct equivalence, when attempting to discover one-to-one semantic correspondences.

We view this problem as closely related to the problem of schema matching in databases [Bellahsene et al., 2011; Bernstein et al., 2011] and, less so, to that of ontology alignment in the Semantic Web [Euzenat et al., 2007; Shvaiko and Euzenat, 2013] and in LD [Jain et al., 2010a; Parundekar et al., 2013]. The focus in this dissertation is kept on managing the uncertainty introduced during the process of discovering such correspondences.
Syntactic vs. Semantic Evidence in Matching. Different matching techniques use different kinds of knowledge as evidence, e.g., element names, string tokens from descriptions, structural information of the position of classes or properties in a hierarchy, semantic knowledge from thesauri or ontologies etc., in their attempt to derive a confidence score for each postulated semantic correspondence. Our BUF is used during the matching stage to assimilate different pieces of evidence and produce a degree of belief for each discovered correspondence. In order to study and contrast it with typical matchers, that derive their decisions by various string-based similarities, the framework is first restricted to assimilating syntactic evidence from local names of construct URIs. At a later stage, terminological knowledge from LD vocabularies is then leveraged to improve the decision making of the matching techniques that work on syntactic evidence alone.

<table>
<thead>
<tr>
<th>Type</th>
<th>ID</th>
<th>Description</th>
<th>Evidence Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntactic knowledge (LE)</td>
<td>SLN</td>
<td>similar-local-name</td>
<td>string similarity($c_S$, $c_T$)</td>
</tr>
<tr>
<td></td>
<td>SL</td>
<td>similar-label</td>
<td>string similarity($c_S$, $c_T$)</td>
</tr>
<tr>
<td></td>
<td>SC</td>
<td>similar-comment</td>
<td>string similarity($c_S$, $c_T$)</td>
</tr>
<tr>
<td></td>
<td>SU</td>
<td>same-URI</td>
<td>string equality(URI_S, URI_T)</td>
</tr>
<tr>
<td></td>
<td>SNS</td>
<td>same-namespace</td>
<td>string equality(NS_S, NS_T)</td>
</tr>
<tr>
<td></td>
<td>SIP</td>
<td>same-immediate-parent</td>
<td>($c_S \subseteq C$) \land ($c_T \subseteq C$)</td>
</tr>
<tr>
<td></td>
<td>SB</td>
<td>subsumed-by</td>
<td>rdfs:subClassOf($c_S$, $c_T$)</td>
</tr>
<tr>
<td></td>
<td>SD</td>
<td>same-domain</td>
<td>($c_S$ rdfs:domain $C$) \land ($c_T$ rdfs:domain $C$)</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>same-range</td>
<td>($c_S$ rdfs:range $C$) \land ($c_T$ rdfs:range $C$)</td>
</tr>
<tr>
<td></td>
<td>SOS</td>
<td>subsumes-and-subsumed</td>
<td>($c_S \subseteq c_T$) \land ($c_T \subseteq c_S$)</td>
</tr>
<tr>
<td>Semantic knowledge (TE)</td>
<td>SA</td>
<td>same-as</td>
<td>owl:sameAs($c_S$, $c_T$)</td>
</tr>
<tr>
<td></td>
<td>EC</td>
<td>equivalent-class</td>
<td>owl:equivalentClass($c_S$, $c_T$)</td>
</tr>
<tr>
<td></td>
<td>EP</td>
<td>equivalent-property</td>
<td>owl:equivalentProperty($c_S$, $c_T$)</td>
</tr>
<tr>
<td></td>
<td>CM</td>
<td>close-match</td>
<td>skos:closeMatch($c_S$, $c_T$)</td>
</tr>
<tr>
<td></td>
<td>EM</td>
<td>exact-match</td>
<td>skos:exactMatch($c_S$, $c_T$)</td>
</tr>
<tr>
<td></td>
<td>DF</td>
<td>different-from</td>
<td>owl:differentFrom($c_S$, $c_T$)</td>
</tr>
<tr>
<td></td>
<td>DW</td>
<td>disjoint-with</td>
<td>owl:disjointWith($c_S$, $c_T$)</td>
</tr>
</tbody>
</table>

Table 4.1: Different kinds of evidence in relation to matching that can be assimilated by the Bayesian Updating Framework.

In more detail, Table 4.1 summarises different kinds of syntactic and semantic knowledge that our framework can make use of. We use the term syntactic evidence to refer to the likelihoods obtained from string-level information, and semantic evidence to refer to the likelihoods derived from various terminological annotations (a.k.a schema knowledge) from LD vocabularies.

The application of the BUF framework in matching is, here, restricted to
one-to-one semantic correspondences between classes and properties from conceptual descriptions. This restriction is motivated by the fact that state-of-the-art schema matching and ontology alignment systems, are mostly limited to discovering one-to-one correspondences [Bernstein et al., 2011]. The flexibility of the BUF framework allows for reasoning over different hypothesis and it can be adapted for deriving degrees of belief for relations other than equivalence.

### Underlying Assumptions

The proposed instantiation of the BUF for matching is based on a probabilistic model that makes assumptions as follows:

- Firstly, we assume that each inferred correspondence is independent of whether any other pair of constructs correspond, conditional on the evidence. Intuitively, this assumption expresses the expectation that the evidence available for each pair of constructs contains sufficient knowledge to derive a correspondence between them.

- Secondly, we assume conditional independence so that the posterior \( P(c_S \equiv c_T | e_1, \ldots, e_n) \) can be computed with Bayesian updating in the light of many observations. This assumption as discussed in Section 4.1.1 allows us to break down the joint conditional probability of the likelihood

\[
P(e_1, \ldots, e_n | H) = \prod_{i=1}^{n} P(e_i | H),
\]

into small manageable parts that we can empirically approximate. Conditional independence allows us to define the conditional probability as the product of the probability of the pairs. Correspondingly, the denominator is factorised to

\[
P(e_1, \ldots, e_n) = \prod_{i=1}^{n} P(e_i).
\]

Deriving a model where all different evidence sources are joined is difficult for the reason that the resulting distribution has a larger sample space which will require much more data to train.
Given the assumptions, Bayesian updating is allowed since Bayes’s theorem can be applied followed by the conditional independence assumption which decomposes the posterior probability into a factor for each evidence. As for the prior, we assume a non-informative prior distribution by adopting the principle of indifference, as already said. The hypothesis of construct equivalence can take one of two states: \( P(H) = \{P(c_S \equiv c_T), P(c_S \not\equiv c_T)\} \) and is, therefore, a special case of discrete hypothesis known as a *Boolean hypothesis*. Thus, for the two possible outcomes our hypothesis can take, \( N = 2 \), the prior probability that one of the outcomes is observed is given by \( 1/N \). In a dataspace setting, we anticipate that there might be cases where the number of equivalent or non-equivalent constructs is known in advance to be higher or lower for a specific state of the hypothesis and in such cases the prior degree of belief might need to be adjusted accordingly.

The probability of the evidence \( P(E) \) can be expressed using the *law of total probability* (Section 4.1.1, Rule 4) as
\[
P(E) = P(E|c_S \equiv c_T)P(c_S \equiv c_T) + P(E|c_S \not\equiv c_T)P(c_S \not\equiv c_T).
\]
For the application of Bayes’s theorem it is essential to estimate the likelihoods for each evidence, i.e., \( P(E|c_S \equiv c_T) \) and \( P(E|c_S \not\equiv c_T) \). For syntactic evidence, our methodology proposes to derive likelihoods for the similarity scores returned by different string-based matchers.

An experiment to empirically derive likelihoods along with an approach to construct the probability distributions for similarity scores is described in Section 4.6. Subsequently in Section 4.6.3 a running example is introduced to illustrate the process of incremental assimilation of syntactic evidence returned as similarity scores from different string-based matchers. Note that the proposed techniques can assimilate any kind of evidence as long as there is a way of constructing the likelihoods.

Section 4.6.1 reports the results of an experiment as to how the likelihoods for each kind of syntactic knowledge can be empirically determined. Section 4.7 describes how semantic knowledge from terminological knowledges from various ontologies can be used as evidence for judging the hypothesis of construct equivalence. To learn the likelihood for each semantic evidence (see Table 4.1), an empirical survey has been conducted by collecting a representative corpus of LD vocabularies from the Web of Data using a crawler. BUF has been integrated as part of the life-cycle of an existing dataspace management system, in particular the Manchester DSToolkit [Hedeler et al., 2012].
4.6 Similarity Scores to Degrees of Belief

Our framework, described in Section 4.5, uses Bayesian updating to assimilate different types of evidence with the aim of discovering semantic correspondences of equivalence. To derive posterior degrees of belief given similarity scores from different string-based matchers (referred to as syntactic evidence) as evidence, it is essential to know the likelihood of observing a similarity score, returned by a matcher, when the hypothesis is true and when the hypothesis is false. Let $\mu$ denote a matcher and $s$ denote a similarity score returned by a matcher for the pair of constructs $\langle c_S, c_T \rangle$. The posterior degree of belief can be expressed using Bayes’s theorem:

$$
P(c_S \equiv c_T | \mu(c_S, c_T) = s) = \frac{P(\mu(c_S, c_T) = s | c_S \equiv c_T) P(c_S \equiv c_T)}{P(\mu(c_S, c_T))},$$  \hspace{1cm} (4.12)

where $P(c_S \equiv c_T)$ is the prior degree of belief on the equivalence of $\langle c_S, c_T \rangle$ and $P(\mu(c_S, c_T) = s | c_S \equiv c_T)$ is the likelihood of observing the evidence when the hypothesis is true (i.e., when constructs are known to be equivalent). As already discussed, the posterior probability depends on how probable the evidence – in this case a similarity score returned from a specific matcher – is in cases where the hypothesis is known to be true (i.e., $c_S \equiv c_T$) and in cases where the hypothesis in known to be false (i.e., $c_S \not\equiv c_T$). This is reflected on the denominator $P(\mu(c_S, c_T))$, which can be expressed as follows:

$$
P(\mu(c_S, c_T)) = s | c_S \equiv c_T) P(c_S \equiv c_T) + P(\mu(c_S, c_T) = s | c_S \not\equiv c_T) P(c_S \not\equiv c_T).$$  \hspace{1cm} (4.13)

The likelihood of observing the evidence when the hypothesis is false i.e., when constructs are known to be non-equivalent ($c_S \not\equiv c_T$) is given by $P(\mu(c_S, c_T) = s | c_S \not\equiv c_T)$. Finally, using the sum rule of probability (see Section 4.1.1), the probability $P(c_S \not\equiv c_T)$ is the complement of $P(c_S \equiv c_T)$, that is

$$
P(c_S \not\equiv c_T) = 1 - P(c_S \equiv c_T).$$  \hspace{1cm} (4.14)

This section now briefly introduces the definitions of the string metrics used by the string-based matchers we have experimented with. The matching algorithms edit-distance and n-gram each use a single metric (for a survey on common string
metrics see [Elmagarmid et al., 2007]) for deriving conclusions on the similarity given a pair of strings over some alphabet \( \Sigma \). Note that the strings are represented as a sequence of letters with no information about their possible meanings.

**Definition 2** (*Normalised Edit-distance*). Given a string \( l \in \Sigma \), an integer \( i \) denotes the index of the \( i \)-th character in \( l \), denoted by \( l[i] \), and \( |l| \) denotes the length of \( n \). The *edit-distance* between a pair of strings \( l_1 \) and \( l_2 \) is the minimum number of insertions, deletions, and substitutions necessary to transform \( l_1 \) to \( l_2 \) denoted by \( ed(l_1, l_2) \). The normalised edit-distance in the interval \([0, 1]\) is derived by \( ed'(l_1, l_2) = ed(l_1, l_2) / \max(|l_1|, |l_2|) \).

**Definition 3** (*Normalised n-gram*). Given \( l_1 \in \Sigma \) and \( l_2 \in \Sigma \), let \((l_1, l_2)\) be a pair of strings and \( ngram(n, l) \) be a function that returns the set of \( n \)-grams, where \( n \) is a positive integer representing the length of a substring \( l \). The similarity between the pair \((l_1, l_2)\) is then determined as \( \text{ng}(l_1, l_2) = |ngram(n, l_1) \cap ngram(n, l_2)| / \min(|l_1|, |l_2|) - 2 \). In what follows we assume that \( n = 3 \), so this particular matcher is referred to as a *trigram* matcher.

Matching algorithms that use a single metric to derive their conclusions are often called *primitive matchers*. Note, however, that it is common for the implementation of the MATCH operator (discussed in Section 2.6) to use multiple primitive matchers to build more sophisticated composite matching techniques [Shvaiko and Euzenat, 2013, 2005; Rahm and Bernstein, 2001]. These matchers can be arbitrary complex and use different techniques. This section argues that the approach used by each matcher (either primitive or more complex) to derive the similarity scores can be ignored once the pattern of its behaviour in returning similarity scores is captured by a probability distribution. For example, matchers often use data model semantics when determining the similarity between constructs (e.g., Cupid [Madhavan et al., 2001] uses the structure of an XML document to support or dispute string-based similarities). Section 4.6.1 introduces an experimental methodology for estimating p.d.f.s for each matcher with the purpose of capturing, with a probabilistic model, how likely it is that a given matcher returns a given similarity score. Regardless of the matching techniques used, the behaviour of each matcher can be captured with a *probabilistic model*.
Definition 4 (Matcher Signature). The similarity scores returned by a matcher can be expressed by two probability distributions over [0, 1]; one for cases where constructs are known to be equivalent, and one for cases where constructs are known to be non-equivalent.

Marie and Gal [2007] showed that the decisions made by a matcher are often masked by noise and uncertain knowledge, which may cause a matcher to be inclined to assign a similarity score of 0 (completely different) for a pair of constructs it conceives not to be equivalent, and a similarity score of 1 (exactly the same) to pairs of constructs it conceives to be equivalent. This bias in decisions can be observed by looking the outcome of a matcher recorded by the similarity matrix. According to [Marie and Gal, 2007], the similarity matrix is sufficient for representing the uncertainty introduced during an automated matching process.

In this section, the uncertainty captured by a similarity matrix is studied using probabilistic models that are then used to perform probabilistic inference when reasoning on the equivalence of a pair of constructs. The matcher signature is used for deriving the likelihoods that are, in turn, necessary for the calculation of posterior probabilities using Bayesian updating.

To derive the likelihoods $P(\mu(c_S, c_T) = s|c_S \equiv c_T)$ and $P(\mu(c_S, c_T) = s|c_S \not\equiv c_T)$, an experimental study has been conducted to observe the similarity scores returned by each of the primitive matchers edit-distance and n-gram with the aim of modelling probability distributions using a kernel estimator.

4.6.1 Empirical Study: Learning from Similarity Scores

This section introduces an experimental methodology that is used to derive probability distributions empirically from similarity scores returned by a matcher.

Although the example matchers considered in this section are limited to the syntactic level, the techniques described in this section are not limited only to string-based matchers. Once the probability distributions are derived from similarity scores, the specific approach followed by each matcher is not relevant to reasoning about the hypotheses.

We assume the existence of a continuous random variable that takes values in the bounded domain [0,1] for the similarity scores, denoted by $S$, with a p.d.f. denoted by $f_S$. The probabilities that can be derived using the density function $f_S$ are used as evidence for judging the hypothesis of equivalence of constructs.
Therefore, this is modelled as a conditional p.d.f. $f_{S|H}$ which can be constructed empirically from sample data. The objective here is to observe the similarity scores $s_i$ for $i \in \{1, ..., n\}$ returned by each matcher $\mu(c_S, c_T) = s_i$ given a fixed hypothesis, i.e., either $H = (c_S \equiv c_T)$ or $H = (c_S \not\equiv c_T)$. In other words, given a fixed hypothesis, the objective is to observe independently the similarity scores returned when constructs are known to be equivalent or non-equivalent.

It is worth mentioning that the random variable for similarity scores is a continuous random variable whereas the hypothesis is represented by a discrete random variable. Thus, the conditional probability distribution function $f_{S|H}$, is of a mixed type. This mixed type probability distribution is the conditional p.d.f. of $S$ given a fixed value of $H$ (see Section 4.3).

As summarised in Table 4.1, evidence from syntactic knowledge in the form of similarity scores returned by the matchers is syntactic in nature. Given a p.d.f. associated with a continuous random variable $S$, the likelihoods can be derived using integrals (see Section 4.2). To demonstrate the experimental methodology used to derive the p.d.f. for each matcher, we use two matching algorithms viz., edit-distance (denoted by $ed$) and $n$-gram (denoted by $ng$). However, the methodology described in this section can be applied to any matcher that returns similarity scores in the range of 0 to 1. For this experiment, the primitive matchers $ed$ and $ng$ are configured to derive string-based similarities over the local names from the URIs given a pair of constructs $(c_S, c_T)$. A local name is the string after the last hash # or slash / of the URI, examples are shown in Figure 4.2.

![Figure 4.2: Examples of URIs with their local-names.](image)

In this case, the experiment is tailored to observing the similarity scores derived by each matcher when matching pairs of local names. The objective is to observe the similarity scores returned by $ed$ or $ng$ and derive their matching signature. The experimental procedure for deriving the probability distributions is depicted in Figure 4.3 as a logical flow diagram.

Recall that for constructing an estimate of the p.d.f.s for each matcher, one
needs to observe the similarity scores returned by each matcher in cases where constructs are known to be equivalent (i.e., the hypothesis is true), and when constructs are known to be non-equivalent (i.e., the hypothesis is false). These similarity scores (from now on referred to as observations) returned by each matcher for each case constitute a sample. We assume that the observations are random and have been obtained independently out of the overall population. The more observations, the more accurate the probability distribution is. Let $G$ denote the set of all observations $\{o_1, \ldots, o_n\}$, where $n$ denotes the sample size. To derive the matcher signature of each matcher, the set of observations $G$ is thought of consisting of the following subsets:

- $E \subseteq G$ contains observations derived when constructs are known to be equivalent i.e., $c_S \equiv c_T$;
- $NE \subseteq G$ contains observations derived when constructs are known to be non-equivalent, $c_S \not\equiv c_T$.

The proposed experimental methodology, therefore, builds on observing the behaviour of each matcher in terms of the similarity scores it returns in various cases. A similar approach has been used for the systematic evaluation of different matching approaches [Do et al., 2002; Bellahsene et al., 2011]. Often, evaluations for matching systems use a collection of test cases (known as benchmarks) with the purpose of evaluating the capabilities of various matching systems. The Ontology Alignment Evaluation Initiative (OAEI)\(^2\) is a community driven initiative

\(^2\)http://oaei.ontologymatching.org
that provides a collection of benchmarks for the systematic evaluation of ontology matching systems [Shvaiko and Euzenat, 2013]. Typically, in such evaluations, different test cases are considered, where each test case involves two ontologies to be matched along with a reference alignment between them (used as the gold standard). Next, each matcher is run on the pair of ontologies to derive an alignment which is compared with the reference alignment provided using different evaluation measures [Euzenat et al., 2011]. Such test cases can be generated automatically by taking one reference ontology and introducing various distortions (or mutations) in a controlled systematic way, e.g., randomly introducing misspellings on the names of classes or properties, introducing different naming conventions, and removing/adding classes or properties. In such cases, the gold standard alignment consists of all the mappings of the original ontology and their mutated versions. A series of systematically generated test cases is made available in the OAEI benchmark track\(^3\).

This section uses a subset of the test cases published in the OAEI benchmark track on which the matchers are run, with the purpose of obtaining the observations that are then used in the construction of the p.d.f.s for each matcher. The experimental methodology (Stage 1 in Figure 4.3) requires a collection of structural descriptions (e.g., schemas, ontologies) as input to the matchers, along with a gold standard. The choice of the OAEI benchmark to provide the test cases for deriving the observations needed for the construction of the p.d.f. seems suitable because (a) it provides a comprehensive benchmark with a variety of test cases that have been generated systematically, (b) the test cases introduce controlled modifications to ontology entities, and (c) it has become the de facto standard for the evaluation of state-of-the-art ontology matching techniques [Euzenat et al., 2011]. Note that the test cases are used here for the purposes of learning the two probability distributions for each matcher: one for when the hypothesis is true, and one for when the hypothesis is false.

**Introducing Systematic Mutations**

Since the matchers `ed` and `ng` considered in this section derive their similarity scores based on the string similarity of local names test cases from the OAEI benchmark track have been chosen that introduce controlled modifications at that level. To satisfy the requirement of Stage 2 in Figure 4.3 of the experimental

\(^3\)http://oaei.ontologymatching.org/2014/benchmarks/
methodology, we have collected around 90 test cases from various OAEI events (ranging from OAEI-2008 to 14) to be used as candidates of deriving observations of similarity scores for our matchers. The selected test cases pair up a construct in the original ontology and its mutated version, generated by a systematic procedure. Such test cases are then used directly as inputs to Stage 3 in Figure 4.3.

In order to ensure that no bias exists on the test cases used, a systematic procedure was implemented to introduce controlled mutations targeting the local names of classes or properties in some ontology (not just the ones that have participated in the OAEI benchmark track). Appendix B gives a detailed description of the different kinds of mutations our systematic procedure introduces along with their probability of occurrence.

Having obtained a collection of test cases from the ones made available under the OAEI benchmark track and having produced additional test cases (around 20 more) by injecting systematic mutations on local names, the next step (Stage 3 in Figure 4.3) is to run the matchers on the local names in each test case and observe the similarity scores returned.

Classifying Similarity Score Observations

Given a test case, let $O$ denote the original version of an ontology, $O'$ denote its mutated version and $A$ their alignment. If $u$ denotes a URI of some construct of an ontology (e.g., see Figure 4.2), which can be either a class or a property definition, then local($\cdot$) is a function which, given $u$, extracts its local name. Let $(l, l')$ denote a pair of local names that has been extracted from a pair of constructs. Each matcher $\mu \in \{\text{ed, ng}\}$ runs independently over the set of all local name pairs $(l, l')$ that belong to the constructs from the input ontologies $(O, O')$. The result of this process is a set of matches. A match is a triple $(l, l', s)$ where $l$ and $l'$ are the local names extracted from the constructs $c_S \in O$ and $c_T \in O'$, respectively, and $s$ is a value in the range of $[0, 1]$ assigning a score to the similarity between $l$ and $l'$ (0: strong dissimilarity; 1: strong similarity).

A diagrammatic representation of the process, with an example of the matches returned using the ed matcher, is depicted in Figure 4.4, which corresponds to Stages 3 and 4 in Figure 4.3.

To obtain the sets $E$ and $NE$, the matches derived by the process (see Figure 4.4) are classified into true positives (TP) and false positives (FP).
CHAPTER 4. MANAGING UNCERTAINTY IN LD MATCHING

Figure 4.4: Example of matches returned by using the edit-distance matcher on pair of local names.

For the example in Figure 4.4, and given the reference alignment for the input ontologies, Figure 4.5 illustrates the classification of the matches derived by the ed matcher. A similar procedure can be followed for ng.

Figure 4.5: Classification of matches to true/false positive matches and the derivation of observations using edit-distance.

Given a classification of the matches, the set of TP matches is used to observe the similarity scores computed by the matcher when matching constructs that are known to be equivalent, which comprise the set \( E \subseteq G \) used in the construction of the p.d.f. \( f_{S|h}(S|h) \) for \( H = (c_S \equiv c_T) \). Similarly, the set of FP matches is used to observe the similarity scores computed by the matcher when matching constructs that are known to be non-equivalent, which comprise the set \( NE \subseteq G \) used in the construction of the p.d.f. \( f_{S|h}(S|\neg h) \) for \( H = (c_S \not\equiv c_T) \).
CHAPTER 4. MANAGING UNCERTAINTY IN LD MATCHING

4.6.2 The Derivation of Score Distributions

To be able to derive degrees of belief for the likelihoods, the conditional p.d.f. $f_{S|H}(S|h)$ and $f_{S|H}(S|\neg h)$ need to be estimated. This section shows how such p.d.f.s can be constructed using the observations in $E$ and $NE$ respectively (Stages 6 to 9 in Figure 4.3). In what follows, it is assumed that the observations of similarity scores from the sets $E$ and $NE$ are independent realisations of the continuous random variable $S$ for each matcher.

Running the matchers ed and ng independently over the 110 test cases collected during the experimental methodology illustrated in Figure 4.3 resulted in 65,671 total number of observations comprising the set $G$, from which 64,991 are elements of $NE \subseteq G$, with the remaining 679 being elements of $E \subseteq G$ for each matcher.

Using the observations of similarity scores from the sets $E$ and $NE$ respectively we now show how the p.d.f., denoted by $\hat{f}$, can be estimated. We begin with a discussion of p.d.f. estimation using histograms. Then, a brief discussion follows on the limitations of parametric approaches in terms of the assumptions that need to be made. Finally, the section concludes by demonstrating how the p.d.f.s for each matcher can be estimated using a non-parametric technique known as kernel density estimation (KDE). For a detailed discussion on KDE the reader is referred to Appendix A.

Using Histograms. Given the set of observations from $E$, Figure 4.6(a) plots a histogram that shows the density (y-axis) of similarity scores. Appendix D shows an example of the data used to plot the histogram, by partitioning the continuous interval $[0, 1]$ of similarity scores (x-axis) into discrete but disjoint bins of equal width. The bin width (denoted by $b$) determines the number of bins and is used to describe the interval of similarity scores that correspond to a single bin. The number of bins has a direct impact on the shape of the histogram [Ioannidis, 2003], as illustrated in Figure 4.6.

The bin width is often identified on a trial-and-error basis [Simonoff, 1996, Ch.2]. Larger bin widths may result in loss of information on the underlying distribution. On the other hand, smaller bin widths result in more bins and more peak characteristics [Wand, 1997]. Determining the optimal bin width is beyond the scope of this dissertation, the reader is referred to [Wand, 1997; Simonoff, 1996, Ch.2] for a discussion.
An empirical estimate of the p.d.f., denoted by $\hat{f}_{S|H}$, can be approximated by a density histogram. Given a similarity score $s \in S$, where $n$ denotes the sample size, the density for that score can be estimated by

$$\hat{f}_{S|H}(s|h) = \frac{1}{n} \times \frac{o_s}{b}.$$  \hspace{1cm} (4.15)

where $o_s$ is the number of observations in the same bin as $s$. Because the entire area in the density histogram sums up to unity, density histograms like the ones in Figure 4.6 can be used to provide empirical estimates of the degrees of belief for the likelihoods required by the BUF framework for the assimilation of similarity scores as syntactic evidence. To derive probabilities, the corresponding density for a similarity score is multiplied by the bin width. However, because histograms tend to suffer from discontinuity [Silverman, 1986, pp.7–11], they are limited in terms of accuracy. This can be demonstrated with a simple example using Figure 4.6(b). Due to lack of observations in the sample, the density for similarity scores in the range (0.04 – 0.12) cannot be computed and it is, therefore, impossible to estimate a probability for a similarity score in that range. In addition, the fact that the shape of histograms is heavily dependent on bin widths influences the Bayesian inference method into providing very different likelihood estimates.

Although there are drawbacks related to their use in density estimation [Silverman, 1986, pp.7–11], histograms are useful as a quick way to validate intuitions on the shape of the p.d.f. For example, we expect that a matcher that is good at matching constructs that are known to be equivalent is more likely to obtain high
similarity scores (closer to 1). In Section 4.6 we discussed that there is uncertainty in the decisions recorded as similarity scores by a matcher. This tendency in decisions can also be observed in the histograms in Figure 4.7(a) and 4.7(b) for the ed matcher and similarly in Figure 4.7(c) and 4.7(d) for the ng matcher. The selection of the bin widths in each case was estimated using Scott’s rule [Scott, 2009, p.48] as implemented in MATLAB.

![Figure 4.7](image.png)

(a) Edit-distance ($b = 0.067$, No. of bins = 15, equivalent case)  
(b) Edit-distance ($b = 0.054$, No. of bins = 17, non-equivalent case)  
(c) Tri-gram ($b = 0.067$, No. of bins = 15, equivalent case)  
(d) Tri-gram ($b = 0.063$, No. of bins = 16, non-equivalent case)

Figure 4.7: Histograms showing the distribution of similarity scores for string-based matchers.

**Using a Parametric Approach.** An alternative technique for density estimation is to use a parametric approach. In brief, this approach assumes that the p.d.f. to be estimated is a member of an existing parametric family of distributions, e.g., the Gaussian with mean $\mu$ and variance $\sigma^2$. The density function can
then be approximated by estimating the parameters $\mu$ and $\sigma$ from the sample of observations at hand. For this approach to work, the shape of the underlying distribution needs to be known beforehand to ensure that the parametric function assumed fits the distribution of the actual observations. As shown by Figure 4.7, the distributions may have different shapes. At a glance, the underlying distributions derived from the observations of similarity scores from each matcher are skewed towards the boundaries 0 and 1, i.e., towards high similarity scores when matching constructs that are known to be equivalent and towards low similarity scores when matching constructs that are known to be non-equivalent. A parametric approach is not suitable for such cases of significant skew. Having shape constraints is the main disadvantage of parametric techniques. In settings where the underlying distributions can take arbitrary shapes incorrect assumptions on the shape of the score distributions limit the benefits of inference techniques [Gray and Moore, 2003].

**Using a Non-Parametric Approach.** We, therefore, use a non-parametric technique for density estimation known as *kernel density estimation*. No assumptions are made about the shape of a distribution but the more data, the better the estimation [Silverman, 1986, Ch.1]. Estimating the p.d.f. for distributions that have arbitrary shape, such as the distributions of similarity scores returned from string-based matchers, using non-parametric techniques seems appropriate [Gray and Moore, 2003].

For the construction of the p.d.f. for the likelihoods we use the *Gaussian kernel*. As for the derivation of the smoothing parameter, routines implemented in MATLAB are used to approximate an optimal value of the parameter using the observed similarity scores for each case.

The continuous random variable $S$ assumed for the similarity scores can only take values from the positive bounded domain $[0, 1]$. Kernel estimators when applied to p.d.f.s in a bounded domain exhibit *boundary effects* [Karunamuni and Zhang, 2008]. This causes density estimations at the boundaries of the domain $[0, 1]$ to be biased. As an example consider Figure 4.8 which plots the estimated $\hat{f}_X$ for the n-gram matcher using the subset of observations $E \subseteq G$ assuming that the kernel estimator applies its smoothing characteristics over the whole real line (i.e., without boundary correction). The kernel assigns densities outside the domain $[0, 1]$, i.e., it assumes that the random variable can take values in
Figure 4.8: Direct application of kernel density estimate over similarity scores derived by the tri-gram matcher \((n = 679, h = 0.038, \text{Gaussian kernel})\) with support \((-\infty, +\infty)\).

Figure 4.9: Boundary corrected kernel density estimate over similarity scores derived by the tri-gram matcher \((n = 679, h = 0.488, \text{Gaussian kernel})\) with support \([0, 1]\).
the entire domain of real numbers. To reduce the effects caused by the direct application of the kernel estimator a simple correction method can be applied based on the transformation methodology proposed by [Marron and Ruppert, 1994]. In our example, the boundary-corrected density estimate is depicted in Figure 4.9. Applying the boundary corrected method causes the estimated p.d.f. to be closer to the boundary region \([0, 1]\) and ensures that the probability densities within the boundary integrates to unity. The corrected p.d.f. minimises the risk of deriving biased probability estimates required for our Bayesian inference framework.

Given the similarity score observations that constitute the sets \(E\) and \(NE\) for each matcher \(\mu \in \{ed, ng\}\) and a kernel estimator (with boundary support), two probability distributions are estimated for each matcher (as shown in Figure 4.10).

As already discussed, the behaviour of each matcher can be represented in terms of the similarity scores it returns, with two probability distributions: one for cases where constructs are known to be equivalent e.g., Figure 4.10(a) and Figure 4.10(c) for the matchers edit-distance and n-gram respectively; and one for cases where constructs are known to be non-equivalent e.g., Figure 4.10(b) and Figure 4.10(d) for the matchers edit-distance and n-gram respectively. The two probability distributions are referred to as the signature of each matcher (Definition 4). It is observed that the p.d.f.s modelled in Figure 4.10 validate the expected rational relationship associated with similarity scores and the hypothesis of equivalence. In other words, high values of similarity scores are more likely to be associated with equivalence than non-equivalence, similarly, low values of similarity scores are more likely to be associated with non-equivalence than equivalence.

Given similarity scores derived from the string-based matchers (i.e., syntactic evidence) as evidence the BUF framework uses the matcher signature to derive the degrees of belief for the likelihood of observing the similarity scores when the hypothesis is true and when it is false. It is then possible to compute the new posterior probability, i.e., for the framework to perform inference.

Note that the methodology is not tied only to string-based matchers that work on local names. Although experiments have not been conducted to construct distributions for other matchers that work on different levels (e.g., on RDF predicates such as \texttt{rdfs:comment} or \texttt{rdfs:label}), the derivation of the distributions for such matcher would follow the same procedure. Moreover, assuming that the sample of observations used for training remains representative, and given that
the behaviour of the matchers $\mu \in \{\text{ed, ng}\}$ is fixed and deterministic, the p.d.f.s derived for each matcher needs not be estimated again. Therefore, the p.d.f.s derived from the principled methodology described throughout the section are derive-once, apply-many constructs. Notice, here that we do not mean that the p.d.f.s for the likelihood need not to be re-estimated ever again, only that if, as should reasonably be expected, the rate of change is slow enough, the cost of one derivation can be amortised over many application episodes. In addition, during the life-cycle of dataspace systems, if more matchers are made available, distributions can be derived provided that a gold standard is given (e.g., from feedback from users on the similarity of constructs during the improvement stage of a dataspace life-cycle).
4.6.3 Updating Degrees of Belief given Similarity Scores

This section demonstrates, with the use of a running example, the application of the Bayesian updating framework to assimilating syntactic evidence returned as similarity scores by string-based matchers.

In order to use the likelihoods of similarity scores returned by each matcher as what we refer to as syntactic evidence, we assume that each piece of evidence (i.e., each similarity score) is conditionally independent given the hypothesis (Section 4.1.1, Rule 5). With the conditional independence assumption, the likelihoods $P(\mu(c_S, c_T) = s|c_S \equiv c_T)$, and $P(\mu(c_S, c_T) = s|c_S \not\equiv c_T)$ for each matcher can be estimated from a sample of similarity score observations using in our case a non-parametric approach, viz., kernel density estimation. Once the likelihoods that describe the signature of each matcher are made available, the outcome of applying the p.d.f.s over a similarity score returned by a matcher is interpreted as the likelihood of that evidence. As an example, let $PDF^+$ denote the p.d.f. for the equivalent case for the $ed$ matcher. Given a pair of local names $(l, l')$, $PDF^+_{ed}(\cdot)$ yields the likelihood that the similarity score $ed(l, l')$ expresses the equivalence of the pair of constructs $(c_S, c_T)$. More formally, $PDF^+_{ed}(ed(l, l')) = P(ed(l, l')|c_S \equiv c_T)$. Similarly, let $PDF^-$ denote the p.d.f. for the non-equivalent case. $PDF^-_{ed}(\cdot)$ yields the likelihood that the similarity score $ed(l, l')$ expresses the non-equivalence of the pair of constructs $(c_S, c_T)$, i.e., $PDF^-_{ed}(ed(l, l')) = P(ed(l, l')|c_S \not\equiv c_T)$. Corresponding definitions exist for the equivalent and non-equivalent case of a similarity score obtained with the $ng$ matcher.

To reason over the hypothesis modelled with a conditional probability, Equation 4.12 can be used. However, note here that the similarity scores returned by each matcher are represented by a continuous random variable; in such a case, as already pointed out, the posterior degree of belief from Equation 4.12 can only be approximated by conditioning on the event that the random variable for similarity scores takes values within a small finite interval $[s - \Delta s \leq S \leq s + \Delta s]$. 

In such a case, Bayes’s rule takes the following form [Bertsekas and Tsitsiklis, 2008]:

\[
P(c_S \equiv c_T|\mu(c_S, c_T) = s) \approx \lim_{\Delta s \to 0} \frac{P(s - \Delta s \leq S \leq s + \Delta s|c_S \equiv c_T) P(c_S \equiv c_T)}{P(s - \Delta s \leq S \leq s + \Delta s)} \\
\approx \frac{\int_{s-\Delta s}^{s+\Delta s} f_E|H(e|h)de P(c_S \equiv c_T)}{\int_{s-\Delta s}^{s+\Delta s} f_E(e)} = \frac{\int f_E|H(e|h)p_H(h)de}{f_E(e)}. \tag{4.16}
\]

As discussed in Section 4.2, the probability that a continuous random variable takes on a specific value is zero, and therefore the posterior probability \( P(c_S \equiv c_T|\mu(c_S, c_T) = s) \) is, instead, conditioned on the event \([s - \Delta s \leq S \leq s + \Delta s]\) where \(\Delta s\) is a small positive number. Equation 4.16, derived above, is actually the mixed form of Bayes’s theorem, where \(f_{H|E}(h|e)\) denotes the conditional p.d.f., which is estimated using kernel estimators. For the continuous part of the mixed distribution, probabilities are calculated as follows:

\[
P(s - \Delta s \leq S \leq s + \Delta s|c_S \equiv c_T) = \int_{s-\Delta s}^{s+\Delta s} f_E|H(e|h) de. \tag{4.17}
\]

For the derivation of the denominator marginal probability \(f_E(e)\), the law of total probability is used:

\[
\sum_{h \in \text{Range}(H)} p_H(h) f_{E|H}(e|h). \tag{4.18}
\]

For completeness, we now discuss, using a running example, how raw similarity scores returned from each matcher can be used to derive degrees of belief for the likelihoods when the hypothesis is fixed, either true or false. Once the degrees of belief for the likelihoods are computed, these can be substituted in Equation 4.16 which calculates the posterior degree of belief. Bayesian updating takes place when a new piece of evidence is obtained (e.g., another similarity score using a different matching process). To be consistent with the matchers ed and ng used
in previous sections, we consider them again here for matching the local names of construct URIs. Figure 4.11 shows a running example on how Bayes’s theorem can be used for the assimilation of different pieces of syntactic evidence.

![Figure 4.11: Bayesian updating with the assimilation of similarity scores returned by the matchers.](image)

The p.d.f.s estimated for each matcher are used to derive the likelihood of observing the similarity score in cases where the pairs of constructs are known to be equivalent or non-equivalent (Figure 4.11, Stages 1–2). The following example shows how one can derive the degree of belief for the likelihood of observing a similarity score \( s = 0.5 \) from edit-distance:

\[
PDF^+_{\text{ed}}(ed(l, l')) = P(ed(l, l') = s \mid c_S \equiv c_T) \\
= P(ed(MusicGroup, Group) = 0.5 \mid c_S \equiv c_T) \\
= P(0.5 - \Delta s \leq S \leq 0.5 + \Delta s \mid c_S \equiv c_T) \\
= \int_{0.5-\Delta s}^{0.5+\Delta s} f_{S|H}(S = 0.5|H = c_S \equiv c_T) \, ds \\
= \int_{0.5-\Delta s}^{0.5+\Delta s} \frac{1}{nh} \sum_{i=1}^{n} K \left( \frac{s - O_i}{h} \right) \, ds \\
= 9.22 \cdot 10^{-3}.
\]

Recall that a kernel estimator is used to estimate the p.d.f. for the similarity scores returned from each matcher where, as always, \( n = 669 \) is the number of
observations (in the equivalent-case \( E \subseteq G \)); \( K \) is the Gaussian kernel function; \( h = 0.67 \), the smoothing parameter; \( O_i \), the series of independent similarity score observations; and \( s = 0.5 \), the value of the continuous random variable for which one seeks to estimate the likelihood. Intuitively, the kernel estimator estimates the density at point \( s = 0.5 \) whereas the integral is used to estimate the probability around the point \( s = 0.5 \) within a small interval, i.e., \([0.5 - \Delta s \leq S \leq 0.5 + \Delta s]\).

This relates back to the p.d.f. in Figure 4.10(a) that shows the density function as a smooth curve which represents the probability distribution of the continuous random variable \( S \). Similarly, the likelihood for the non-equivalent case (with observations from \( NE \subseteq G \)) using edit-distance is derived as follows:

\[
PDF_{ed}^{-\Xi}(ed(l, l')) = P(ed(l, l') = s \mid c_S \neq c_T) \\
= P(ed(MusicGroup, Group) = 0.5 \mid c_S \neq c_T) \\
= P(0.5 - \Delta s \leq S \leq 0.5 + \Delta s \mid c_S \neq c_T) \\
= \int_{0.5 - \Delta s}^{0.5 + \Delta s} f_{S \mid H}(S = 0.5 \mid H = c_S \neq c_T) \, ds \\
= \int_{0.5 - \Delta s}^{0.5 + \Delta s} \frac{1}{nh} \sum_{i=1}^{n} K \left( \frac{s - O_i}{h} \right) \, ds \\
= 5.04 \times 10^{-3}.
\]

In the non-equivalent case, the number of independent similarity score observations \( n = 64,991 \), \( K \) is a Gaussian kernel function as in the equivalent case; \( h = 0.649 \), the smoothing parameter; \( O_i \), the series of independent similarity score observations; and \( s = 0.5 \), the value of the continuous random variable which one seeks to estimate. Correspondingly, the same logic is applied for the equivalent and non-equivalent cases for the ng matcher.

To reason over our hypothesis of construct equivalence, Bayes’s theorem is therefore used to assimilate the syntactic similarity score observed using \( ed \) and then to revise the computed posterior degree of belief in the light of new evidence. In the running example, the new evidence is the similarity score observed using the ng matcher. The Bayesian updating occurs when the posterior degree of belief, calculated with the similarity score returned by \( ed \), assumes the role of the prior in the computation of the revised posterior (Figure 4.11, Stage 3). To conclude our running example, the computation and revision of the posterior
decreases of belief is as follows:

\[ P(c_S \equiv c_T|ed(MusicGroup, Group) = 0.5) \approx \lim_{\Delta s \to 0} P(c_S \equiv c_T|0.5 - \Delta s \leq S \leq 0.5 + \Delta s) \]

\[ \approx \frac{\int_{0.5-\Delta s}^{0.5+\Delta s} f_{S|H}(S = 0.5|H = c_S \equiv c_T) ds}{\int_{0.5-\Delta s}^{0.5+\Delta s} f_{S|H}(S = 0.5|H = c_S \equiv c_T) ds} \]

A revision of the posterior follows in the light of new evidence,

\[ P(c_S \equiv c_T|ng(MusicGroup, Group) = 0.5454) \approx \lim_{\Delta s \to 0} P(c_S \equiv c_T|0.5454 - \Delta s \leq S \leq 0.5454 + \Delta s) \]

\[ \approx \frac{\int_{0.5454-\Delta s}^{0.5454+\Delta s} f_{S|H}(S = 0.5454|H = c_S \equiv c_T) ds}{\int_{0.5454-\Delta s}^{0.5454+\Delta s} f_{S|H}(S = 0.5454|H = c_S \equiv c_T) ds} \]

Discussion

Section 4.5 discussed on Bayes’s theorem ability of learning to adjust a subjective probability for a hypothesis to reflect the latest available evidence. In the first round of Bayesian updating and assuming a uniform prior \( P(c_S \equiv c_T) = 0.5 \), the running example showed that the hypothesis, given as evidence the similarity score from ed increases with respect to the prior degree of belief. The impact of the evidence is observed by the relationship \( P(ed(l, l') = 0.5|c_S \equiv c_T) > P(ed(l, l') = 0.5) \) where the likelihood \( P(ed(l, l') = 0.5|c_S \equiv c_T) = 9.22e-3 \) is greater than the denominator 7.13e-3. This causes a confirmation (or increase) of the posterior degree of belief. In the second round of Bayesian updating the calculated posterior \( P(c_S \equiv c_T | ed(l, l') = 0.5) = 0.64 \) from the previous round assumes the role of the prior, which with the impact of the new evidence from ng results to a further increase of the posterior to \( P(c_S \equiv c_T | ng(l, l') = 0.5454) = 0.77 \). Similarly, the impact of the similarity score (i.e., the evidence) received from ng causes a further confirmation of the hypothesis. This indeed can be observed by the computed likelihood \( P/ng(l, l') = 0.5454 | c_S \equiv c_T) = 1.52E-2 \) which is greater than the
denominator $1.26\times10^{-2}$ which is confirming the hypothesis in terms of the prior.

To further understand the effect of each piece of syntactic evidence we studied them independently from each other. Given the p.d.f.s for the likelihoods empirically estimated for each matcher, and assuming a series of prior probabilities in the range $[0, 1]$, Figure 4.12 plots the posterior degree of belief for each combination. In particular, Figure 4.12(a) plots the posteriors for judging the hypothesis when the evidence is similarity scores returned by the ed matcher i.e., $P(c_s \equiv c_T \mid ed(c_s, c_T) = s)$. Similarly, Figure 4.12(b) plots the posteriors when evidence is similarity scores returned by the ng matcher i.e., $P(c_s \equiv c_T \mid ng(c_s, c_T) = s)$.

![Figure 4.12: Effect on the posterior probabilities for syntactic evidence assuming different priors.](image)

Moreover, plots in Figure 4.12 allow one to observe the behaviour of the matchers. For example, the edit-distance matcher causes an increase in the posterior when it returns a similarity score in the range of $[0 - 0.4]$ given some prior. In contrast for n-gram matcher a similarity score in the range of $[0 - 0.4]$ is more trustworthy towards indicating whether pairs are syntactically likely to be equivalent.

To conclude, this section illustrated the application of the BUF framework for the assimilation of different pieces of syntactic evidence for different matchers. Given the likelihoods constructed using kernel estimators from Section 4.6.2, the framework performs Bayesian updating by first translating raw similarity scores to degrees of belief using the p.d.f.s derived and then computes the posterior degree of belief that shows the confidence on the hypothesis, i.e., whether the a pair of constructs is equivalent given the available evidence. Previously, we discussed that the BUF framework can be used for the assimilation of different pieces of
evidence once the likelihoods for that evidence become available. This flexibility of the framework for incrementally revising a posterior degree of belief for a hypothesis offers the opportunity to bring together different pieces of evidence.

4.7 Semantic Annotations to Degrees of Belief

This section explores the intuition that the semantic annotations in LD vocabularies can be leveraged to mitigate the shortcomings introduced by matching processes that are based on syntactic evidence alone.

Typical matching techniques proposed as solutions to the schema matching [Rahm and Bernstein, 2001] or ontology alignment [Shvaiko and Euzenat, 2013] problems, often aggregate different pieces of knowledge (i.e., syntactic, structural, background knowledge etc.) to make judgements on construct equivalence. Based on the idea of combining different pieces of knowledge, we have proposed, earlier in this chapter, the use of a BUF for assimilating different pieces of evidence, and we have shown how the framework can be instantiated for schema matching. The fact that LD resources are annotated with semantically rich knowledge using shared ontologies presents an opportunity to utilise such semantic annotations to improve the degrees of belief obtained from purely syntactic evidence from the matching algorithms traditionally used.

As summarised in Table 4.1 in Section 4.5, semantic annotations from LD vocabularies can be classified under the following categories: (a) syntactic knowledge attached to an ontology entity, such as labels (e.g., rdfs:label, skos:prefLabel) and comments (e.g., rdfs:comment) in the form of text, (b) semantic knowledge such as relations between entities, either internal to a vocabulary or across different LD vocabularies, e.g., subsumption (is-a), as used to model class or property hierarchies and equivalence, as owl:equivalentClass assertions.

Syntactic knowledge that comes from the values of predicates like rdfs:label, rdfs:comment or skos:prefLabel (see Figure 4.13 for an example) can be assimilated by our framework once the estimated probability distributions for the similarity scores returned by matchers are empirically derived. Although not demonstrated here, a string-based or token-based matcher that works on that level would follow a similar procedure for initialisation and evidence assimilation to the one described in Section 4.6. For example, consider a hypothetical matcher that computes the similarity of a pair of constructs by considering the string (or text)
values of different predicates (e.g., label, comments etc.) as a bag of words. Such a matcher can use techniques from information retrieval (e.g., cosine similarity) to derive similarity scores based on a construing the bag of words as a vector of strings and assuming a metric space [Euzenat et al., 2007]. To construct the likelihoods needed for the assimilation of similarity scores returned by the matcher, we would derive its matcher signature by constructing two probability distributions from observing the similarity scores returned by the matcher when judging the hypothesis of construct equivalence when constructs are known to be equivalent (positive case) and non-equivalent (negative case), using the methodology described in Section 4.6.

In addition to syntactic knowledge, LD vocabularies contain rich semantic knowledge. We consider semantic knowledge for example, RDFS/OWL annotations asserting subsumption or equivalence relations between classes or properties.

This section proposes the use of such semantic knowledge to improve the decision making of matching processes that otherwise depend on syntactic knowledge alone. For example, Figure 4.13 depicts a pair of classes that share the same immediate direct parent (a relationship we denote by SIP) using RDFS constructs (e.g., rdfs:subClassOf). Such terminological knowledge can be used as additional evidence to refine the decision making of matchers that only consider syntactic evidence for judging the hypothesis of construct equivalence.

Figure 4.13: Pieces of syntactic and semantic knowledge at the conceptual level between a pair of RDFS/OWL classes.
To be able to reason over the hypothesis of construct equivalence using as evidence different kinds of semantic knowledge, it is essential to derive the likelihoods of observing the kind of semantic knowledge in cases where the hypothesis holds (i.e., constructs are known to be equivalent) and in cases where the hypothesis does not hold (i.e., constructs are known to be non-equivalent). Let $\lambda$ denote a piece of semantic knowledge that holds between a pair of constructs e.g., $\lambda(c_S, c_T) = SIP$. An update of the prior degree of belief in the light of evidence that constructs $(c_S, c_T)$ share the same immediate parent can be computed using Bayes’s theorem. In particular, another form of Equation 4.12 is:

$$P(c_S \equiv c_T | \lambda(c_S, c_T) = SIP) = \frac{P(\lambda(c_S, c_T) = SIP | c_S \equiv c_T)P(c_S \equiv c_T)}{P(\lambda(c_S, c_T) = SIP)}.$$ (4.19)

This is similar to reasoning over the hypothesis of construct equivalence using as evidence similarity scores from matchers. Bayes’s theorem is used by the BUF framework to calculate or revise a posterior degree of belief to reflect the available evidence, in this case the knowledge of a SIP relationship. Correspondingly, $P(c_S \equiv c_T)$ is the prior degree of belief for the hypothesis, $P(\lambda(c_S, c_T) = SIP | c_S \equiv c_T)$ is the likelihood of observing the evidence given the hypothesis, and $P(\lambda(c_S, c_T) = SIP)$ is the degree of belief of the evidence itself, which can be derived, as before, using the law of total probability. As already discussed, to be able to compute the posterior of Equation 4.19 it is crucial that the likelihoods of observing the evidence when the hypothesis is true e.g., $P(\lambda(c_S, c_T) = SIP | c_S \equiv c_T)$, and when the hypothesis is known to be false $P(\lambda(c_S, c_T) = SIP | c_S \neq c_T)$, are known.

However, the question remains as to how much weight to assign on each piece of semantic knowledge. The intuition is that if a piece of semantic knowledge (e.g., SIP) has an impact on the hypothesis of equivalence it should occur more often in cases where constructs are known to be equivalent than in cases where constructs are known to be non-equivalent. The trustworthiness of each piece of semantic knowledge for judging the equivalence of a pair of constructs (i.e., our hypothesis) can be derived through an empirical study. The following section describes a survey conducted over a corpus of LD vocabularies collected from the WoD. The corpus was used for empirically estimating the likelihoods for each piece of semantic knowledge.
4.7.1 Empirical Study: Learning from the Web of Data

For constructing likelihoods from semantic annotations retrieved from the Web of Data – referred in this dissertation as semantic evidence – an up-to-date representative sample of the conceptual level (i.e., T-Box) of the WoD was collected.

We first review related attempts to obtain empirical samples from the WoD that motivated our studies. Our aim is to have a well-defined methodology for retrieving a representative sample from the WoD at the conceptual level to support an empirical derivation of the likelihoods necessary for assimilating evidence from semantic annotations. A first attempt to crawl the WoD was attempted for the Billion Triple Challenge (BTC) [Bizer and Maynard, 2011] using the MultiCrawler framework [Harth et al., 2006] which provided an empirical sample of the state of the WoD. A seed list used as input to a crawler was built from URIs retrieved from various semantic web search engines (e.g., SWSE [Hogan et al., 2011]). In a complementary approach, [Käfer et al., 2012;Käfer et al., 2013] describe an experiment for observing the dynamicity of the WoD and a framework for collecting frequent snapshots using sampling. For retrieving a representative sample of the WoD, a seed list of URIs is extracted from the LOD Cloud project\(^4\) and the BTC 2011 dataset [Harth, 2011]. The seed list obtained is then used as input to a crawler that performs periodical crawling over the WoD with the aim of obtaining a corpus of LD documents.

For the purposes of our experiment, we are interested in the perspective of the WoD focusing on the conceptual level. Such a perspective is available under the Linked Open Vocabularies (LOV) project\(^5\), which maintains a collection of LD vocabularies used for the publication of RDF datasets in the LOD cloud\(^6\). In addition, the LODStats project [Auer et al., 2012] maintains various statistics for RDF datasets published under the Comprehensive Knowledge Archive (CKAN) project\(^7\). Besides the statistical analysis provided, the LODStats project maintains a list of the most frequently used LD vocabularies. Other approaches to crawl the WoD have been attempted by various semantic web search engines such as SWSE [Hogan et al., 2011] and Swoogle [Ding et al., 2004].

To the best of our knowledge, none of the samples retrieved focuses only on the conceptual level. A possible candidate sample which might be suitable for our

\(^4\)http://thedatahub.org/group/lodcloud
\(^5\)http://lov.okfn.org/dataset/lov
\(^6\)http://linkeddata.org/
\(^7\)http://thedatahub.org
purposes is the one made available under the “LOV Aggregator”\(^8\) feature of the LOV project. This dataset is made available under a single SPARQL endpoint with a dump file available for download. Although this feature provides a best-effort (remote SPARQL endpoints are unstable), up-to-date version of the listed vocabularies, it does not seem to offer a complete coverage of the LD vocabularies used on the whole of the WoD. It is difficult to assert how representative the composition of that dataset is of the WoD, at the conceptual level. It is important that the sample used for empirically estimating the likelihoods for the different pieces of semantic annotations (see Table 4.1) is representative. As discussed by Käfer et al. [2012], perhaps a clear picture of the WoD can be provided by crawlers that harvest RDF from the WoD; the LOV project only covers a prominent subset of the LD vocabularies.

This section now describes: (a) a principled methodology for obtaining a representative sample of the WoD at the conceptual level by taking a series of repeated samples; and (b) the use of the repeated samples to approximate the probability distributions for each kind of semantic annotation, given the hypotheses of construct equivalence and non-equivalence. To reduce the error when estimating the probability distributions, the proposed methodology relies on the use of the central limit theorem [Bertsekas and Tsitsiklis, 2008, Ch.7].

### Sampling Methodology

Due to the potentially large corpus of LD vocabularies available on the WoD, it is necessary to sample in order to ensure an appropriate collection of LD vocabularies that can be processed for empirically deriving the likelihoods for semantic evidence under the normal, coverage hardware, time and bandwidth restrictions. The goal of our sampling methodology is therefore to obtain a representative corpus of LD vocabularies published on the WoD that (i) span a wide range of domains and (ii) can be used for empirically deriving the likelihoods for semantic annotations given the available computing infrastructure.

From our brief discussion of initiatives to crawl the WoD, we conclude that existing sampling methods ensure a spread across different domains by composing a seed list that spans different hosts. The selection of the URIs that compose the seed list is performed either randomly, by selecting either URIs from semantic web search engines [Harth et al., 2006] or registered URIs to services like the

---

\(^8\)http://lov.okfn.org/dataset/lov/agg/
LOD “cloud”, or else by ranking URIs from existing WoD samples (e.g., the BTD dataset 2011) based on a PageRank analysis and selecting the most popular ones [Käfer et al., 2012]. The reader is referred to a survey of crawling approaches in the Web of documents [Bennouas and de Montgolfier, 2007].

![Diagram](image)

Figure 4.14: A lightweight model for random crawling the Web of Data.

In a nutshell, Figure 4.14 illustrates an overview of the methodology used for obtaining multiple random samples from the WoD with a focus on the conceptual level.

1. **Seed list.** Since our major requirement is to obtain samples from the WoD that target the conceptual level, it is reasonable to use URIs that are registered under the LOV project and those collected from popular LD vocabularies populated under the LODStats project. Since both projects collect URIs from various domains in the ecosystem of published LD vocabularies from various parts of the WoD, this gives some guarantee that the samples cover popular domains. A total of 786 URIs were collected\(^9\) from both projects to comprise the initial seed list used as input to the proposed crawling methodology.

2. **Random selection of seed URIs.** Recall that our purpose in obtaining a sample from the WoD is to estimate the probability distributions for each kind of semantic annotation. As we have no knowledge of the distribution of each semantic annotation, the target is to derive an empirical approximation of the distribution using repeated random sampling [Helton et al., 2006] over the collected population of LD vocabularies. At this stage, we are

\(^9\)Retrieved on 15/06/2014
only interested in obtaining the population of vocabularies by retrieving multiple samples from the WoD. For each round, the methodology selects at random (without replacement) a percentage of URIs to crawl from the total list of URIs (i.e., the given seed list). We configured the methodology to randomly select 50% of the URIs for each round, with all URIs having equal probability of being selected.

3. **LD crawler.** The crawling configuration is essential to ensure that our samples span multiple domains. This allows us to observe the frequency of occurrence of each semantic annotation from the retrieved samples. For the crawling part of the methodology introduced in Figure 4.14, the implementation used LDSpider [Isele et al., 2010b], an open-source multi-threaded crawling framework designed for LD. In more detail, LDSpider was configured as follows:

- **RDF syntaxes:** Most popular LD vocabularies are serialised using RDF/XML or Turtle syntax; crawling was restricted only to consider such vocabularies.
- **Load-balance:** Given the randomly selected seed list the crawler was configured to fetch 500K URIs. This configuration ensures that the fetched LD documents are distributed between the domains.
- **Link filter:** The crawler has been restricted into following only T-Box links to ensure that our samples remain at the conceptual level.
- **Multi-threading:** multi-threading was enabled to ensure load-balancing between CPU and disk [Käfer et al., 2012].
- **Output:** The output from each round is stored in a triple store using named graphs, one for each sample with the use of SPARQL/Update. The Jena TDB triple store was used with Fuseki as the SPARQL server.

4. **Shuffle seed URIs.** To ensure randomness of the samples retrieved as well as diversity of the LD vocabularies crawled, the URIs from the given seed list are shuffled at the end of each round. In some respects, this is similar to the strategy used for crawling the BTC 2011 dataset [Harth, 2011].

---

Due to the unpredictability of the remote connections, the crawling experiment was repeated for 20 rounds using the initial seed list of 786 URIs. At the end of the crawling experiment, our approximation of a representative corpus of the population of LD vocabularies from the WoD resulted in 4.5 million N-Quad statements. Figure 4.15 shows the number of N-Quads retrieved at each crawling round. On average 220K N-Quads were retrieved at each round, with some discrepancies in cases where the crawling failed mostly due to the unpredictability of the remote connections.

![Figure 4.15: Number of N-Quads retrieved per crawl round.](image)

**Brief Survey of the Corpus**

Before elaborating on the derivation of the likelihoods and to further understand the amount of terminological knowledge collected in our corpus of LD vocabularies, we now present an empirical survey of popular language constructs adopted on the WoD. The survey focuses on the usage of language constructs from well-known vocabularies such as RDFS/OWL and SKOS. Figure 4.16 gives a visual overview of the listed terms from each vocabulary. The y-axis shows the number of distinct subject URIs for each language feature shown in the x-axis. It seems that most vocabularies use OWL 1 language elements that are restricted to the expressiveness of the OWL DL profile. The OWL DL profile puts restrictions on
the use of RDF(S) rdfs:Class and rdf:Property language constructs, the declaration of classes using these terms is less numerous than the OWL versions of owl:Class, owl:ObjectProperty and owl:DatatypeProperty. In terms of the features of the SKOS vocabulary, skos:Concept is the most heavily used term in relation to other features from the same vocabulary, such as skos:exactMatch, skos:broadMatch and skos:narrowMatch.

From the recently proposed OWL 2 language features, the most prominently used feature is owl:NamedIndividual. The OWL 2 features with the least usage in LD vocabularies collected in our corpus are owl:AllDisjointClasses, owl:AllDisjointProperties, owl:OntologyProperty, owl:ReflexiveProperty, and owl:IrreflexiveProperty which is the least used feature. On the other hand, the most prominent term used is owl:Class, followed by owl:unionOf and owl:Restriction. From our brief survey of the crawled corpus one can identify several modelling errors. For example, there are 90 occurrences of owl:subClassOf and 6 occurrences of owl:subPropertyOf, which do not exist as language constructs in OWL. Additionally, we found some inconsistencies in the use of some language constructs. For example, owl:sameAs and owl:differentFrom are intended to be used at the instance level to indicate equivalent/disjoint URIs that denote the same/different individuals and not on the conceptual level to indicate equivalent/disjoint Classes/Properties.

For completeness we mention here that the above survey finds similarities with the work discussed in [Glimm et al., 2012], during an approach to gain insights of which OWL fragment is currently adopted by publishers on the WoD. For our purposes the survey conducted provides insights for the kinds of assertions that can be used for training the likelihoods necessary for the assimilation of evidence, as construed by the Bayesian approach described throughout this chapter.

To sum up, the above survey on the occurrence of language features used in LD vocabularies is appropriate for our present purposes but is only based on a best-effort approximation and should not be considered as definitive. The next section will elaborate on how the semantics of language features from RDF-S/OWL and SKOS are used to build input sets comprising assertions that can be construed as direct/indirect evidence of equivalence or non-equivalence of constructs (that are either classes or properties). For example, the semantics of the OWL language features owl:equivalentClass and owl:equivalentProperty are a direct assertion that a pair of Classes/Properties is equivalent. Correspondingly,
Figure 4.16: Vocabulary elements of RDFS/OWL and SKOS.
owl:disjointWith and owl:propertyDisjointWith assert that a pair of Classes/Properties are non-equivalent.

### 4.7.2 The Derivation of Mass Functions

This section describes a methodology for learning the likelihoods for each type of semantic knowledge in Table 4.1 using the retrieved corpus of LD vocabularies. Firstly, we assume the existence of a Boolean random variable, with domain \{true, false\}, for each kind of semantic knowledge from the set \(TE = \{SIP, SB, EC, EP, EM, CM, DW, SD, SR\}\). Since the Boolean variable is a special kind of discrete random variable, the probability distribution for each semantic knowledge can be modelled with a p.m.f., denoted by \(p_{E|H}\), which can be estimated empirically from sample data. The objective here is to empirically observe how often pair of constructs (either classes or properties) are related by a relationship in \(TE\) given a fixed value of the hypothesis, either \(H = (c_S \equiv c_T)\) or \(H = (c_S \not\equiv c_T)\). As an example, consider a pair of constructs that share the same immediate parent, as evidence of semantic knowledge, i.e., \(\lambda\langle c_S, c_T\rangle = SIP\). If we denote by \(|SIP \cap \equiv|\) the number of pairs of constructs that have both SIP and \(\equiv\) relationships in the sample and by \(|\equiv|\) the number of \(\equiv\) relationships there are, then the p.m.f. \(p_{E|H}\) when \(H = (c_S \equiv c_T)\) can be empirically approximated as follows:

\[
p_{E|H}(e|h) = P(\lambda\langle c_S, c_T\rangle = SIP | c_S \equiv c_T) = \frac{P(SIP, c_S \equiv c_T)}{P(c_S \equiv c_T)} = \frac{|SIP \cap \equiv|}{|\equiv|}. \tag{4.20}
\]

To empirically estimate the likelihoods, as in the example above, pairs of classes and properties were collected from the corpus of LD vocabularies crawled from the WoD. The pairs are organised into two homogeneous sets using direct and indirect assertions of equivalence/non-equivalence using language features from RDFS/OWL and SKOS vocabularies. The sets of equivalence/non-equivalence pairs are defined as follows:

- \(EE\) is the set of pairs of classes/properties that are asserted to be equivalent using the semantics (or inference semantics) of language features from RDFS/OWL and SKOS, summarised in Table 4.2.
• NE is the set of pairs of classes/properties that are asserted to be non-equivalent using the semantics (or inference semantics) of language features from RDFS/OWL and SKOS, summarised in Table 4.3.

To express the hypothesis of construct equivalence/non-equivalence the language features in Tables 4.2 and 4.3 were used to infer a degree of equivalence/non-equivalence between pairs of classes or properties. The features selected for the empirical derivation are not definitive: other features used to annotate classes/properties with strict or a variant degree of semantic equivalence/non-equivalence can be used. For the purposes of deriving the likelihoods for each type of semantic knowledge, the approach is to combine the language features and establish the empirical basis upon which the derivation takes place. The pairs of classes/properties for each set are collected using SPARQL queries constructed for each suggested rule (Appendix F shows examples of the kinds of SPARQL queries used). As an example, a collection of candidate pairs of classes for the set EE (i.e., that of equivalent classes) is obtained with the use of a SPARQL Query as in Listing 4.1.

```
SELECT DISTINCT ?c1 ?c2
WHERE {
    {?c1 a rdfs:Class .}
    UNION {?c1 a owl:Class .}

    FILTER (!isBlank(?c2) && ?c1 != owl:Nothing
    && ?c2 != owl:Nothing && ?c1 != owl:Thing
    && ?c2 != owl:Thing)
}
```

Listing 4.1: Pair of Classes that are asserted to be equivalent.

In brief, the set EE comprises 10,149 pairs of classes that are asserted to be equivalent, whereas the set NE for non-equivalent pairs of classes comprises 4,040 pairs. For completeness we consider properties, although classes are sufficient for exploring whether semantic evidence can be leveraged for improving the decision making of matchers that mostly work on the syntactic level. Thus, correspondingly, the set of EE for properties comprised of 9,837 pairs of equivalent properties, and the set of NE comprised 8,104 pairs of non-equivalent properties.
### Table 4.2: Rules used to derive the set of equivalent pairs for classes/properties.

<table>
<thead>
<tr>
<th>ID</th>
<th>Assertion</th>
<th>Implies</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS-EQU</td>
<td>?c₁ owl:equivalentClass ?c₂</td>
<td>equivalence of classes</td>
</tr>
<tr>
<td>CLS-SUB</td>
<td>?c₁ rdfs:subClassOf ?c₂ ?c₂ rdfs:subClassOf ?c₁</td>
<td>equivalence of classes</td>
</tr>
<tr>
<td>CLS-EM</td>
<td>?c₁ skos:exactMatch ?c₂</td>
<td>high degree of confidence that classes can be used interchangeably</td>
</tr>
<tr>
<td>CLS-CM</td>
<td>?c₁ skos:closeMatch ?c₂</td>
<td>sufficiently similar that classes can be used interchangeably</td>
</tr>
<tr>
<td>PRP-EQU</td>
<td>?p₁ owl:equivalentProperty ?p₂</td>
<td>equivalence of properties</td>
</tr>
<tr>
<td>PRP-SUB</td>
<td>?c₁ rdfs:subPropertyOf ?c₂ ?c₂ rdfs:subPropertyOf ?c₁</td>
<td>equivalence of properties</td>
</tr>
<tr>
<td>PRP-EM</td>
<td>?p₁ skos:exactMatch ?p₂</td>
<td>high degree of confidence that properties can be used interchangeably</td>
</tr>
<tr>
<td>PRP-CM</td>
<td>?p₁ skos:closeMatch ?p₂</td>
<td>sufficiently similar that properties can be used interchangeably</td>
</tr>
<tr>
<td>PRP-INV</td>
<td>?p₁ owl:inverseOf ?p₂</td>
<td>property is the inverse of some other property</td>
</tr>
</tbody>
</table>

### Table 4.3: Rules used to derive the set of non-equivalent pairs for classes/properties.

<table>
<thead>
<tr>
<th>ID</th>
<th>Assertion</th>
<th>Implies</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS-DW</td>
<td>?c₁ owl:disjointWith ?c₂</td>
<td>classes have no common members in their extensions</td>
</tr>
<tr>
<td>CLS-CO</td>
<td>?c₁ owl:complementOf ?c₂</td>
<td>equivalent to using owl:disjointWith</td>
</tr>
<tr>
<td>CLS-AD</td>
<td>?C rdf:type owl:AllDisjointClasses</td>
<td>collection of classes that are mutually disjoint</td>
</tr>
<tr>
<td>PRP-DW1</td>
<td>?p₁ owl:propertyDisjointWith ?p₂</td>
<td>properties are disjoint</td>
</tr>
<tr>
<td>PRP-AD</td>
<td>?P rdf:type owl:AllDisjointProperties</td>
<td>collection of properties that are mutually disjoint</td>
</tr>
<tr>
<td>PRP-DW2</td>
<td>{?p₁ rdfs:range ?rng₂ .} ∧ {?p₂ rdfs:range ?rng₂ .} ∧ {?rng₁ owl:disjointWith ?rng₂ .}</td>
<td>properties are disjoint</td>
</tr>
</tbody>
</table>
From subsets of equivalent/non-equivalent classes or properties, the probability for each kind of semantic knowledge is estimated by repeatedly retrieving random samples from the sets $EE$ and $NE$, respectively, for classes and properties. Then for each sample retrieved, the probability of each kind of semantic evidence is computed (using the technique in Equation 4.20).

To understand the quality of the estimates of each conditional probability, we use the *central limit theorem* [Bertsekas and Tsitsiklis, 2008, Ch.7]. The central limit theorem assumes that the distribution of the sample mean is a normal distribution and allows us to have a notion of the accuracy of the estimated probabilities using the standard error. As an example, to estimate the likelihood $P(\lambda(c_S, c_T) = SIP \mid c_S \equiv c_T)$, we proceeded as follows.

1. Random sampling was performed over the set $EE$ to retrieve a sample of $N$ equivalent pairs of classes.

2. From the sample, the number of equivalent pairs of classes that share the same immediate parent was counted.

3. The likelihood is empirically estimated (using Laplace correction for smoothing).

The mean conditional probability estimated for different semantic evidence is shown in Table 4.4.

<table>
<thead>
<tr>
<th>evidence</th>
<th>mean ($\mu$)</th>
<th>sigma ($\sigma$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNS</td>
<td>0.84812</td>
<td>0.01257</td>
</tr>
<tr>
<td>SIP</td>
<td>0.00712</td>
<td>0.00336</td>
</tr>
<tr>
<td>SB</td>
<td>0.84120</td>
<td>0.01303</td>
</tr>
<tr>
<td>EC</td>
<td>0.94656</td>
<td>0.00792</td>
</tr>
<tr>
<td>EM</td>
<td>0.03752</td>
<td>0.00804</td>
</tr>
<tr>
<td>CM</td>
<td>0.01308</td>
<td>0.00473</td>
</tr>
<tr>
<td>DW</td>
<td>0.00199</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.4: Mean conditional probabilities assigned to each semantic evidence (equivalent) classes.

Similarly, the conditional probabilities estimated for the non-equivalence case for classes, e.g., $P(\lambda(c_S, c_T) = SIP \mid c_S \not\equiv c_T)$ are shown in Table 4.5.

We consider the empirically derived conditional probabilities for each semantic evidence as *derive-once, apply-many* constructs, however, since the vocabulary
### Table 4.5: Mean conditional probabilities assigned to each semantic evidence (non-equivalent) classes.

<table>
<thead>
<tr>
<th>evidence</th>
<th>mean ($\mu$)</th>
<th>sigma ($\sigma$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNS</td>
<td>0.96156</td>
<td>0.00885</td>
</tr>
<tr>
<td>SIP</td>
<td>0.49788</td>
<td>0.01997</td>
</tr>
<tr>
<td>SB</td>
<td>0.44120</td>
<td>0.02063</td>
</tr>
<tr>
<td>EC</td>
<td>0.00199</td>
<td>0</td>
</tr>
<tr>
<td>EM</td>
<td>0.00199</td>
<td>0</td>
</tr>
<tr>
<td>CM</td>
<td>0.00199</td>
<td>0</td>
</tr>
<tr>
<td>DW</td>
<td>0.06036</td>
<td>0.02207</td>
</tr>
</tbody>
</table>

collection from which we have drawn our sample is dynamic, we might wish to view them as derive-seldom, apply-often.

The following section demonstrates with an example, the application of the framework to assimilating semantic evidence as construed by the BUF (see Table 4.1).

#### 4.7.3 Updating Degrees of Belief given Semantic Evidence

As an example, let $PMF^+$ denote the p.m.f. for the equivalent case. Given a pair of classes $(c_S, c_T)$, $PMF^+_SIP(\cdot) = P(SIP(c_S, c_T)|c_S \equiv c_T)$. More formally, $PMF^+_SIP = SIP(c_S, c_T)) = P(SIP(c_S, c_T)|c_S \equiv c_T)$. Similarly, let $PMF^-$ denote the p.m.f. for the non-equivalent case. $PMF^-_SIP(\cdot) = P(SIP(c_S, c_T)|c_S \neq c_T)$. Corresponding definitions exists for the equivalent and non-equivalent case for each of the kinds of semantic evidence made use by the BUF.

To reason over the hypothesis of construct equivalence in the case of having observed that $(c_S, c_T)$ have a SIP, the Bayes’s rule takes the following form:

$$P(c_S \equiv c_T|\lambda(c_S, c_T) = SIP) = \frac{P(\lambda(c_S, c_T) = SIP|c_S \equiv c_T) P(c_S \equiv c_T)}{P(\lambda(c_S, c_T) = SIP)} = \frac{p_{E|H}(e|h)p_H(h)}{p_E(e)}.$$  \hfill (4.21)

As already stated, $p_{E|H}(e|h)$ is the conditional p.m.f. A conditional probability
can then empirically calculated for the equivalent case, this is shown in Table 4.4 and for the non-equivalent, this is shown in Table 4.5. For the derivation of the denominator $p_E(e)$, the marginal probability is derived using the law of total probability:

$$
\sum_{h \in \text{Range}(H)} p_H(h)p_{E|H}(e|h).
$$

(4.22)

For completeness, and for understanding the confidence assigned to different semantic evidence, we studied their independent behaviours analytically using Bayes’s theorem. Given the empirically derived likelihoods, Figure 4.17 shows how the posterior is updated in the presence of a semantic evidence, given a series of prior probabilities.

Figure 4.17: Examples of posterior probabilities computed for different semantic evidence.
For example, Figure 4.17(a) shows how the posterior is influenced by observing that a pair of classes belongs to the same namespace (i.e., SNS). Based on our intuitions this should not play much difference in their judgement of equivalence.

In another case, Figure 4.17(b) shows how the judgement on the equivalence is influenced if there is direct evidence that a pair of classes share the same immediate parent (i.e., SIP). Our approach interprets this evidence as an indication of their disjointness and therefore, the posterior is significantly decreased. From the other hand, a subsumption relationship (i.e., SB) may indicate that the constructs are more likely to be related than disjoint and a low prior is slightly increased, this is observed in Figure 4.17(c). Finally, Figure 4.17(d) shows how the posterior is affected when a pair of constructs stand in an equivalence relationship (i.e., EC). This evidence is considered enough evidence to significantly increase a low prior to 1 meaning that constructs are more probably equivalent than if that evidence had not been available.

Note that similar observations can be derived for all the semantic evidence that the BUF can understand. However, the ones mentioned are considered sufficient for understanding our contributions.

4.8 Experimental Evaluation

This section, presents our experimental studies to evaluate the effectiveness of the Bayesian updating framework in uniformly managing the uncertain decisions inherent in the matching stage of the dataspace life-cycle. The aim of our experimental studies is to provide a principled treatment of uncertainty in relation to the evidence introduced while matching constructs, but in a way that can be replicated in other phases of the life-cycle.

Given the lack of a benchmark for measuring uncertainty in data integration processes like matching, the evaluation presented here is based on the idea of emulating construct equivalent judgements produced by human experts when presented with different kinds of syntactic and semantic evidence. The experimental studies have a twofold purpose: (i) to compare how well the principled assimilation of syntactic evidence alone, with the use of Bayesian updating, performs against the aggregation of syntactic evidence with a predefined function; specifically average (AVG)\textsuperscript{13} and (ii) to observe whether the incorporation of semantic evidence

\textsuperscript{13}The use of AVG as a strategy for aggregating the decisions of matchers is considered here
evidence improves treatment to the decisions regarding construct equivalence that are based on syntactic evidence alone.

4.8.1 Experimental Setup

The empirical evaluation presented in this section considers the following:

Matchers. Primitive matchers \textit{edit-distance} (denoted by \texttt{ed}) and \textit{n-gram} (denoted by \texttt{ng}) are used for computing the syntactic similarities given pairs of local name strings. The primitive matchers focus on the derivation of one-to-one semantic correspondences between pair of constructs. In addition, and to preserve uncertainty in their decisions, the experimental evaluation does not make use of a filtering strategy for the matchers (e.g., a threshold).

Assumptions. The empirical evaluation assumes the following: (i) a non-informative prior by adopting the principle of indifference; (ii) the different kinds of evidence are independent from each other; (iii) the probability distributions for the likelihoods for each evidence were derived once, and independently from each other.

Precedence. In iterative processes, the order in which information is presented may be important. For Bayesian updating, a previously calculated posterior assumes the role of the next prior, applied iteratively in the light of new evidence. It can be proved (see Appendix C) that the final result of Bayesian updating on a set of observations is independent of the sequence order in which each observation is presented.

Survey Study

The experimental evaluation is grounded on judgements made by human experts. A group of human experts was asked to make judgements on the equivalence of a pair of constructs as participants to a survey study. More specifically, human experts were asked to judge whether a pair of constructs is postulated to be equivalent in the presence of different kinds of evidence, both syntactic and semantic (as construed in this chapter).

as the \textit{baseline}. This is honourable since it is a common aggregation strategy commonly used by existing matching techniques [Aumüller et al., 2005; Bernstein et al., 2011].
Participants. The survey was distributed and completed by 15 human participants all experts in solving data integration tasks, such as schema matching and mapping. The questions were tailored in a way so as to capture the confidence of people when asked to judge the hypothesis of construct-equivalence when presented with different pieces of syntactic and semantic evidence. An example question is shown in Figure 4.18.

Q.1. Given two schema elements that:
   a. their local-name strings share no similarity.
   b. belong to the same URI namespace.
   c. a subsumption relationship exists between them.

For example, `mo:Medium` and `mo:CD`, such that `mo:CD` is a subclass of `mo:Medium`.

State how confident you are that the pair of elements is equivalent:
- Definitely equivalent
- Tending towards being equivalent
- Do not know
- Tending towards being not equivalent
- Definitely not equivalent

Figure 4.18: Example of a survey question.

For the survey study, a set of pairs of constructs (i.e., classes and properties) from different LD vocabularies was collected, ensuring that different combinations of syntactic and semantic evidence were present or absent. By observing different pairs of constructs from the collected ontologies, approximately 40 common combinations of syntactic and semantic evidence have been identified. For each combination, a question was designed to obtain individual testimonies from each responder.

Testimonies were recorded for each question using a discretisation scale [de Vaus, 2002], as follows: {Definitely equivalent} mapped to a degree of belief of 1.0; {Tending towards being equivalent} mapped to a degree of belief of 0.75; {Do not know} mapped to a degree of belief of 0.5; {Tending towards being not-equivalent} mapped to a degree of belief of 0.25; and {Definitely not-equivalent} mapped to a degree of belief of 0. For each question, individual testimonies were then aggregated using a weighted average. The aggregated degrees of belief obtained from the survey are treated as an approximation of the experts’ confidence on construct equivalence, given certain pieces of syntactic and semantic evidence and act as the gold standard for the experimental studies discussed later on.
CHAPTER 4. MANAGING UNCERTAINTY IN LD MATCHING

Expectation Matrix

Given the set of constructs from $S$, the set of constructs from $T$ and the set of syntactic and semantic evidence available for each pair $\langle c_S, c_T \rangle$, where $c_S \in S$ and $c_T \in T$, a degree of belief is assigned for each pair as derived from the experts’ testimonies. Let $M_{exp}$ be a $n \times m$ structure, referred from now on as the expectation matrix, where $n = |S|$ and $m = |T|$. The element $e_{jk}$ in the $j$-th row and the $k$-th column of $M_{exp}$, denotes the degree of belief derived from the survey between the $j$-th construct in $S$ and the $k$-th construct in $T$, according to the available evidence.

Evaluation Metric

Let $p_1, p_2, \ldots, p_n$ be the degrees of belief derived for a pair of constructs using either the AVG aggregation scheme or the Bayesian scheme, and let $a_1, a_2, \ldots, a_n$ be the corresponding degrees of belief in the expectation matrix. The mean absolute error (MAE) is used to measure the aggregated error, denoted by $\delta$, for all the individual $|p_i - a_i|$ errors. Given $n$ total numbers of individual errors, then:

$$MAE = \frac{|p_1 - a_1| + \ldots + |p_n - a_n|}{n}.$$ (4.23)

In the literature, there are alternative ways for comparing a set of predicted values with their eventual outcomes. Through empirical evaluation the choice of MAE seemed the most appropriate, essentially because it does not exaggerate on the effect of outliers, in comparison to most of the alternatives [Hyndman and Koehler, 2006].

Baseline

To compare the results from the evaluation, the experimental scenario considers a baseline, motivated by a common strategy for aggregating the independent results derived by different matching techniques. The error derived using AVG is considered to be the baseline error and is derived as follows. Given a pair of conceptual descriptions $S$ and $T$, an antagonist to the Bayesian updating methodology is a process that independently runs the string-based matchers ed and ng on the local names of constructs. The similarity scores derived from each matcher (construed as judgements of the matchers’ degree of belief on construct equivalence) for each
pair of classes are aggregated using AVG. The aggregated result of this process is a matrix, denoted by $M_{\text{avg}}$. The baseline error is obtained by computing how close the predictions in $M_{\text{avg}}$ are to those in $M_{\text{exp}}$, i.e., the degrees of belief obtained by the experts’ testimonies. The MAE (from Equation 4.23) is used as the numeric prediction performance measure. The aggregated error between the results obtained with using the AVG aggregation strategy $M_{\text{avg}}$ and the results from the expectation matrix $M_{\text{exp}}$, is denoted by $\delta_{\text{avg}}$ (referred here as the baseline error).

### 4.8.2 Experimental Description

The purpose of the experimental scenario described in this section is to measure any improvements in the decision making of the string-based matchers when semantic evidence is made explicit use of. To simulate the kind of cross-links that are likely to be published, as RDF statements across LD vocabularies, this scenario considers that a set of vocabulary links is made available by BLOOMS\(^{14}\), which is a technique for discovering `rdfs:subClassOf` and `owl:equivalentClass` semantic relations between LD vocabularies [Jain et al., 2010a]. It is worth noting that the Bayesian methodology for assimilating different pieces of evidence is independent of external techniques such as BLOOMS.

**Datasets.** This experimental scenario uses a pair of ontologies made available in the Ontology Alignment Evaluation Initiative (OAEI) benchmark, which is targeted to ontology alignment tasks. In particular, the reported results considered a pair of ontologies, viz., *ekaw* and *sofsem*, that have been developed in the OntoFarm project [Chen and Nugent, 2011] and used in the conference track of the OAEI competition\(^{15}\). These ontologies belong to the domain of conference organisation and have been developed independently from each other, therefore, share no cross-links between them.

\(^{14}\)To ensure vocabulary links of sufficient quality, BLOOMS was configured with a high threshold, viz., 0.8.

\(^{15}\)http://oaei.ontologymatching.org/2014/conference
CHAPTER 4. MANAGING UNCERTAINTY IN LD MATCHING

Table 4.6: Participating ontologies from OAEI conference track.

<table>
<thead>
<tr>
<th>name</th>
<th># of classes</th>
<th># of matches (edit-distance)</th>
<th># of matches (n-Gram)</th>
<th>bayesian technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>ekaw</td>
<td>74</td>
<td>4227</td>
<td>911</td>
<td>4389</td>
</tr>
<tr>
<td>sofsem</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Methodology. This scenario considers the pair of ontologies ekaw and sofsem as the conceptual descriptions provided as input to the process, respectively denoted by $S$ and $T$. Similarly to the previous experiment the baseline error is computed from aggregating the similarity scores (using average) derived for the local names of classes from the ontologies. The Bayesian assimilation technique was then used (as the aggregation technique instead of an average) to assimilate syntactic evidence computed by the string-based matchers $ed$ and $ng$. The result of this process is a $n \times m$ matrix, denoted by $M_{syn}$, where $n = |S|$ and $m = |T|$. The element $e_{jk}$ in the $j$-th row and the $k$-th column of $M_{syn}$, denotes the posterior probability, i.e., the degree of belief on construct equivalence between the $j$-th class in $S$ and the $k$-th class in $T$, according to the syntactic evidence made available by the string-based matchers. The next step is to measure how close the predictions from $M_{syn}$ (using syntactic evidence alone) are to the expectation matrix $M_{exp}$. The result is denoted by $\delta_{syn}$.

To assess how valid the hypothesis of using the available semantic evidence to provide improved treatment to the uncertain decisions inherent in judging construct equivalence on the basis of syntactic evidence alone, this scenario first uses BLOOMS to make explicit the cross-ontology semantic relations and then uses them as the available semantic evidence. In the light of this new evidence, the Bayesian assimilation technique updates the posterior probabilities for each pair of classes from $M_{syn}$ accordingly. The result of this process is a new matrix $M_{syn,sem}$ with the same dimensions as $M_{syn}$, where the posterior probabilities for the elements $e_{jk}$ reflect both syntactic and semantic evidence available. Again, the error between the revised posteriors is calculated between $M_{syn,sem}$ and the expectation matrix $M_{exp}$, the computed result is denoted by $\delta_{syn,sem}$.

Finally and to complete the evaluation, we also examine the individual absolute errors used for computing the mean absolute errors: $\delta_{avg}$, $\delta_{syn}$, and $\delta_{syn,sem}$. The results of this experimental scenario are now discussed more thoroughly.
Results and Discussion

Baseline vs. Bayesian Syntactic. The mean absolute error computed for the baseline aggregation scheme using average against the expectation matrix was $\delta_{\text{avg}} = 0.1079$, whereas, the error resulted from the process of assimilating syntactic evidence alone using the Bayesian techniques was $\delta_{\text{syn}} = 0.0698$. The difference of 0.0381 between the errors $\delta_{\text{avg}}$ and $\delta_{\text{syn}}$ can be expressed in percentage terms as 35.32%. A detailed look at the individual errors calculated for each aggregation scheme reveal a better improvement on the Bayesian technique. More specifically and to further understand the difference in errors, we observed the absolute errors that fall into each of four regions of interest as these are depicted in Figure 4.19.

Figure 4.19: Regions of interest used for evaluation.

The regions of interest roughly correspond to the following minimum bounding rectangles, resp., Region 1 lies below the $y = x$ error line where AVG error $>>$ Bayesian error and is the rectangle defined by $y = 0.2$; Region 2 lies above the $y = x$ error line where AVG error $<<$ Bayesian error and is the rectangle defined by $x = 0.2$; Region 3 lies below the $y = x$ error line where AVG error $>$ Bayesian error and is the rectangle defined by $y > 0.2$; and Region 4 lies above the $y = x$ error line where AVG error $<$ Bayesian error and is the rectangle defined by $x > 0.2$. 
To support the hypothesis that the Bayesian technique makes better judgements on the hypothesis of construct equivalence, in the light of different evidence, we expect most of the individual absolute errors calculated to fall in Region 1, which we construe as the golden zone, since, inside it, our technique performs better, i.e., has smaller individual errors. In particular, for the baseline aggregation scheme that produced \( M_{\text{avg}} \) we counted 3,833 matches with individual absolute errors greater than the analogous individual errors derived by the Bayesian technique that produced \( M_{\text{syn}} \). The use of the Bayesian technique for aggregating syntactic evidence significantly outperformed (i.e., has much smaller individual errors than) the use of the AVG as the aggregation scheme for syntactic evidence for 87.49% of the matches. This observation topped our expectations. Our principled treatment of uncertainty with the use of the developed Bayesian methodology was shown to improve the decision making of the matching stage when compared with a state-of-the-art aggregation scheme. Table 4.7 summarises the results according to the regions of interest with a count on the individual absolute errors located in each region, in both absolute terms and relative to the total.

<table>
<thead>
<tr>
<th>no. region</th>
<th>count</th>
<th>perc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ( R_{\text{avg}&gt;&gt;B_{\text{syn}}} )</td>
<td>3833</td>
<td>87.49</td>
</tr>
<tr>
<td>2 ( R_{\text{avg}&lt;&lt;B_{\text{syn}}} )</td>
<td>215</td>
<td>4.90</td>
</tr>
<tr>
<td>3 ( R_{\text{avg}&gt;B_{\text{syn}}} )</td>
<td>31</td>
<td>0.70</td>
</tr>
<tr>
<td>4 ( R_{\text{avg}&lt;B_{\text{syn}}} )</td>
<td>302</td>
<td>6.89</td>
</tr>
</tbody>
</table>

Table 4.7: AVG scheme vs. Bayesian syntactic.

**Baseline vs. Bayesian Syn. & Sem.** To evaluate our hypothesis as to whether semantic annotations from LD vocabularies can improve the uncertain decisions of string-based matchers, this experiment compares the errors \( \delta_{\text{avg}} \) and \( \delta_{\text{syn,sem}} \). The mean absolute error \( \delta_{\text{syn,sem}} = 0.1259 \) is lower than \( \delta_{\text{avg}} = 0.1942 \) with a difference of 0.0683 or 35.15%. The improvements on the individual absolute errors are observed on pairs of classes that have some semantic evidence, indicating that the existence of a semantic relation has a positive impact on reducing its uncertainty. From a total of 4,381 one-to-one matches discovered 175 of them share a semantic relation. Figure 4.20 plots the individual absolute errors calculated for such cases. We are interested on the individual errors where the Bayesian technique delivers much smaller errors than the AVG scheme.

In particular, the Bayesian technique significantly outperforms (i.e., has much
smaller error) mostly between 0.1 and 0.3 with a very few cases almost as high as 0.75 on the x-axis, and below the \( y = x \) error line. A 71.43% of the 175 cases of matches derived that share a semantic relation most individual errors lie in the golden zone (i.e., Region 1). This improvement coincides with our expectations showing that the existence of semantic evidence causes individual decisions to lie closer to the expectations of human experts. In some cases, however, the Bayesian technique is outperformed (i.e., has larger error) mostly with individual errors that lie in Region 2 for 24.57% of the cases. With respect to the total number of individual cases, this percentage is relatively small and therefore on average the Bayesian technique (considering the available semantic evidence) delivers improved results. Table 4.8 shows in detail the results for each region showing a count of the individual errors located in each of the regions of interest, in both absolute terms and relative to the total.

<table>
<thead>
<tr>
<th>no.</th>
<th>region</th>
<th>count</th>
<th>perc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( R_{avg} &gt;&gt; B_{syn,sem} )</td>
<td>125</td>
<td>71.43</td>
</tr>
<tr>
<td>2</td>
<td>( R_{avg} &lt;&lt; B_{syn,sem} )</td>
<td>43</td>
<td>24.57</td>
</tr>
<tr>
<td>3</td>
<td>( R_{avg} &gt; B_{syn,sem} )</td>
<td>2</td>
<td>1.14</td>
</tr>
<tr>
<td>4</td>
<td>( R_{avg} &lt; B_{syn,sem} )</td>
<td>5</td>
<td>2.85</td>
</tr>
</tbody>
</table>

Table 4.8: AVG scheme vs. Bayesian syntactic & semantic.
Bayesian Syn. vs. Bayesian Syn. & Sem. This case compares the Bayesian technique when assimilates syntactic evidence alone (denoted by $B_{syn}$) against the configuration of the Bayesian technique that assimilates both syntactic and semantic evidence available (denoted by $B_{syn,sem}$). This experiment compares the errors $\delta_{syn}$ and $\delta_{syn,sem}$. With respect to the previous experiment we observed a similar behaviour that further coincides with our expectations that, in general the semantic evidence available can improve the results to a significant degree. In more detail we observed that, $\delta_{syn,sem} = 0.1259$ which is closer to the expectation matrix than $\delta_{syn} = 0.2768$ with a difference of 0.1509 or 54.5%. Detailed results are summarised in Table 4.9 and depicted as individual absolute errors for each case in Figure 4.21.

![Figure 4.21: Individual absolute errors showing Bayesian syntactic vs. Bayesian syntactic and semantic.](image)

The use of semantic evidence in $B_{syn,sem}$ significantly reduce the individual errors for 89.21% of the total matches discovered, in comparison to the $B_{syn}$ that utilises syntactic evidence alone. In particular, $B_{syn,sem}$ significantly outperforms (i.e., has much smaller individual errors, with almost all falling in the golden zone between 0.2 and 0.3) the use of syntactic evidence alone i.e., $B_{syn}$. Very few individual error cases show that the assimilation of syntactic evidence alone i.e., $B_{syn}$ outperforms (i.e., has much smaller error) the use of semantic evidence i.e., $B_{syn,sem}$. Finally, some individual errors remain unchanged (i.e., lie on the
y = x error line) showing no improvement between the different schemes (i.e., $B_{syn}$ and $B_{syn,sem}$).

Overall, it seems that the incorporation of the semantic relations discovered as semantic evidence help to improve the decision making between pairs of classes that otherwise the combination of string-based matchers was producing more uncertain results. In general, all experiments showed that the Bayesian technique is perhaps an adequate way for combining the results of various heterogeneous matchers with semantic evidence using a uniform and a principled way.

<table>
<thead>
<tr>
<th>no. region</th>
<th>count</th>
<th>perc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{B_{syn}&gt;&gt;B_{syn,sem}}$</td>
<td>124</td>
<td>89.21</td>
</tr>
<tr>
<td>$R_{B_{syn}&lt;&lt;B_{syn,sem}}$</td>
<td>9</td>
<td>6.48</td>
</tr>
<tr>
<td>$R_{B_{syn}&gt;B_{syn,sem}}$</td>
<td>5</td>
<td>3.60</td>
</tr>
<tr>
<td>$R_{B_{syn}&lt;B_{syn,sem}}$</td>
<td>1</td>
<td>0.72</td>
</tr>
</tbody>
</table>

Table 4.9: Bayesian syntactic vs. Bayesian syntactic & semantic.

Conclusions

From the evaluation studies, we concluded that the scarcity of vocabulary links between LD vocabularies have a less positive effect on our methodologies at the same time restricting the scope of the kinds of semantic evidence (as construed in this chapter) to be assimilated. However, our experimental scenario topped our expectations, proving that when such knowledge bases are enriched with semantic relations, the Bayesian technique for assimilating evidence improves significantly the quality of decision making for string-based matching techniques. The reduction of uncertainty at the schema stage is beneficial for several reasons previously discussed in this dissertation over the entire life-cycle of a dataspase.

In addition, the experimental evaluation showed that the Bayesian assimilation of syntactic evidence when used as an alternative aggregation strategy (instead of an average), in all cases delivers better judgements of construct equivalence compared with the average aggregation strategy used by most matching techniques.

The experimental studies revealed that the Bayesian methodology performs well in terms of reducing the uncertain decisions while matching constructs in a setting where cross-ontology links are made explicit. In the future, we expect
that the WoD will be more integrated at the conceptual level, with more semantic knowledge becoming available to be leveraged by our methodology.

4.9 Related Work

Proposals for schema matching [Bernstein et al., 2011] and for ontology alignment [Shvaiko and Euzenat, 2013] typically derive their results by combining together different pieces of evidence by which they revise their confidence with respect to the similarity between the available concepts. This strategy has proved successful. This chapter took the view that uncertainty in decision making at the matching stage of these attempts can be represented by probability distributions. The work described in [Marie and Gal, 2007] also uses probability distributions to model the tendency in matching results but differs in important respect. Unlike in our contribution, a parametric model based on the beta distribution is used to model matcher behaviour, whereas our methodology uses a non-parametric approach based on kernel estimators.

Also, we additionally contribute a principled methodology for assimilating different pieces of evidence in relation to matching, viz., Bayesian updating. The proposal by Marie and Gal [2007] uses heuristics based on a naïve Bayes classifier in the form of features. In another approach, Eckert et al. [2009] propose the use of different kinds of information about the nature of the elements to be matched in order to learn rules for the correctness of a correspondence using classifiers. As additional features to the syntactic evidence, our methodology takes as evidence semantic annotations from LD vocabularies that are published on the WoD.

In an alternative work on matching query interfaces across multiple Web databases, Hong et al. [2010] proposed the use of an evidential approach to combine multiple pieces of evidence using evidential reasoning [Lowrance and Garvey, 1982]. More specifically, Dempster-Shafer theory of evidence [Shafer, 1976] is used to reason with uncertain and incomplete knowledge with the purpose of matching query interfaces. This approach of evidential reasoning could be considered similar to the contribution described in this chapter from the sense that different pieces of evidence are used to confirm or dis-confirm a given hypothesis in relation to matching. Also, unlike traditional matching approaches, that apply different weight coefficients to the match results from individual matchers with the purpose of aggregating their results, the techniques described by [Hong et al., 2010] require
no tuning or weight coefficients. This is, in a way, similar to our contribution of using Bayesian updating for reasoning the hypothesis of construct equivalence using different pieces of evidence which also requires no tuning for aggregating the different kinds of evidence, assuming that the probability distributions are made available for each evidence in advance.

OMEN [Mitra et al., 2005] is a complementary approach that builds on a Bayesian Network for mapping ontologies. Similarly to the methodology described in this chapter, OMEN associates probabilities to each pair of concepts from the ontologies. In addition, OMEN builds a Bayes Net based on a set of so called \textit{meta-rules} to capture the influence of the ontology structure. Our developed methodology captures cross-ontology relations and assigns weights (derived from estimated probability mass functions) to capture their influence on the probabilities derived from syntactic evidence alone.

Zhang et al. [2013] assume the existence of a probabilistic set of schema matchings and propose the use of crowdsourcing to reduce the uncertainty of schema matching that is inherent in the inability of schemas to capture the semantics of the data. Individual correspondences of pair of attributes are used to formulate questions that can be published to the crowd as simple human intelligence tasks. Shannon entropy is used for representing uncertainty in each question by assigning probabilities which are revised accordingly to the confirmations from the crowd. Although, our methodology does not uses the crowd to confirm or disconfirm the derived correspondences, it can provide a set of probabilistic schema matches that can be used as input to their described techniques. Additionally, the use of Bayesian updating as described in this chapter can be used to incrementally update the posterior probabilities according to evidence from the crowd.

The most similar work to the methodology presented in this chapter is the work by [Mao, 2013], which also considers an approach for deriving probabilities from similarity scores returned by matchers. However, the approach described in that study uses empirical counts on a curve that captures the trend of \textit{true positive} matches over the total number of matchers (respectively for the \textit{false positive} case), instead of a proper probability distribution that is learned from similarity scores returned from each matcher. Additionally, [Mao, 2013] illustrates with a scenario study how the use of user feedback can be used as additional pieces of evidence to the uncertain decisions of matchers. In contrast, our methodology
assigns weights (represented as probabilities) on several semantic annotations used to annotate the data and utilises them to improve the decision making of matchers that build on the syntactic evidence alone.

[Lin et al., 2011] uses RDF datasets to build predictive models using machine learning algorithms. Specifically, the authors show how to learn relational Bayesian classifiers with the use of statistical SPARQL queries. The use of SPARQL queries is somewhat relevant to our approach used to learn the probability distributions for different pieces of semantic evidence.

4.10 Discussion and Conclusions

Automatic techniques used for discovering associations between the schemas, and for the derivation of semantic mappings are often faced with some level of uncertainty into their decisions. This chapter discussed on a methodology for reasoning under uncertainty using Bayesian updating as a technique. The applicability of the developed techniques has been demonstrated through a study over judging the equivalence of constructs; a common conclusion that matching techniques are required to reason upon during the matching stage. The decisions made by the proposed methodology are represented as subjective degrees of belief, which are updated in the light of different pieces of evidence; either syntactic from similarity scores returned by various matchers or semantic from knowledge that comes from LD vocabularies.

The experimental studies were grounded on the judgements of human experts regarding construct equivalence when syntactic and semantic evidence is available. Two main results have been obtained: Bayesian techniques can be used for the assimilation of different kinds of evidence in relation to matching, and when semantic annotations are present the decision making of matchers that typically work on syntax only is improved.
Chapter 5

Concluding Remarks

“You have your way. I have my way. As for the right way, the correct way, and the only way, it does not exist.”

Friedrich Nietzsche

This chapter presents a review of the research results contributed in this dissertation and discusses their significance in Section 5.1. Section 5.2 discusses some limitations and future directions that arise in this context.

5.1 Significance of Results and Contributions

This section summarises the research contributions resulting from the research described in this dissertation, in relation with the research objectives set in Section 1.4 and discusses their significance.

Traditional data integration has proved cost-ineffective for settings such as the on-demand-integration of sources at Web scale. The WoD has been characterised by many as a large dataspace [Hausenblas and Karnstedt, 2010; Heath and Bizer, 2011; Umbrich et al., 2012] where the participants (i.e., LD datasets) are published in an open setting by various publishers, and where the semantic integration between LD datasets is treated mostly with a linkage approach, based on manual efforts by the publishers. The emergence of the WoD as a dataspace is still ongoing research and has a very active community. It seems clear that various benefits from pay-as-you-go data integration can be utilised to provide treatment of the various kinds of heterogeneities introduced on LD datasets, at the same
time, providing a coherent and integrated view of the data that will, eventually, enable query answering.

This motivated objective O1 (to identify the challenges and opportunities stemming from an attempt to apply pay-as-you-go data integration techniques over LD). The contribution associated with this objective provided an account, through a case-study as described in Section 2.6.1, of the identified challenges and opportunities in LD integration when the principles and techniques from pay-as-you-go data integration (as these are envisioned by a dataspace) are applied over LD data sources. The case-study was significant in pointing out the research challenges and opportunities for applying data integration techniques (e.g., identification of matches, derivation of mappings) and principles (i.e., incremental improvement in the form of payment) to the integration of LD. It also pointed out that there was an opportunity for the semantic annotations in LD datasets to be used in reducing the uncertainty inherent in dataspaces due to the extensive use of automation throughout the life-cycle phases. Although not all challenges and opportunities resulting from objective O1 are explored in this dissertation, our exploration raised interesting research questions.

One question that this dissertation has addressed is how to deal with the “schema-less” nature of LD datasets? Data integration processes, rely on knowledge of the schemas of the sources. The lack of an explicit conceptual description over LD sources became an important topic for further exploration given the overall aim of exploring how dataspace techniques can be applied to LD. This observation let to objective O2 (to devise techniques that overcome the problems that prevent LD sources from being treated as classical sources from the point of view of the dataspace life-cycle). To support the automatic initialisation of a dataspace, the techniques to stem from objective O2 needed to be automatic and highly autonomous. The resulting contribution is the design and implementation of a methodology that infers a conceptual structure for RDF graphs. To satisfy the requirement of automatically inferring conceptual summaries (so as to support the pay-as-you-go style of integration envisioned by dataspaces), the approach builds on cluster analysis. Chapter 3 described the developed methodology in detail along with an experimental evaluation.

The need for a conceptual level description of LD datasets has been also observed by [Jain et al., 2010b; Hausenblas and Karnstedt, 2010; Paton et al., 2012] and in research on federated query processing over LD (as described in
Section 2.5.1). Although some efforts have been made to devise a solution [Quilitz and Leser, 2008; Alexander et al., 2009] these approaches focus more on statistical aspects of the datasets and do not provide a clear conceptual structure that can enable data integration and query formulation (more on Chapter 3). It is significant because such conceptual summaries can support data integration and query formulation, as well as, saving the user from the need of issuing exploratory SPARQL queries for a manual exploration of the source.

Another interesting research question that arises from the explorations stemming from objective O1 is how to model and provide a systematic treatment of uncertainty in dataspaces in a way that it can be propagated through the life-cycle phases? Although this research question is quite broad and needs extensive further exploration, well beyond the scope of this dissertation, our contributions with regards to objective O3 (to develop techniques that take advantage of the rich semantic annotations used to describe LD sources in order to improve the outcomes of the bootstrapping phase in the dataspace life-cycle) resulted in:

(i) a principled methodology for representing and quantifying the uncertainty resulting from string-based matchers and from the use of semantic evidence in LD vocabularies in a uniform way (i.e., as subjective probabilities);

(ii) a principled approach to evidence assimilation of both syntactic and semantic evidence using Bayesian updating that leads to significant improvements in the accuracy of construct-equivalence hypotheses when compared with conventional assimilation techniques (e.g., average) used in the state-of-the-art matching platforms.

The significance of these results stems from the opportunities arising in terms of managing uncertainty in every phase of the life-cycle of a dataspace. The Bayesian approach to evidence assimilation opens up opportunities for a systematic treatment of uncertainty, as well as solutions to the many challenges discussed in Section 2.6.2 in relation to uncertainty management in dataspaces.
CHAPTER 5. CONCLUDING REMARKS

5.2 Limitations and Future Research

5.2.1 Soliciting User Feedback

The reliance on user feedback at different stages of the life-cycle phases is a crucial assumption in a dataspace management system for improving the quality of the speculative integrations that resulted from the extensive use of automation at the bootstrapping phase.

The Bayesian approach to evidence assimilation opens up opportunities for acquiring feedback on the derived matches as pieces of evidence [Mao, 2013], thus enabling their assimilation in a useful way in terms of reducing the uncertainty introduced during the matching phase. Essentially, our Bayesian methodology can, in principle, be used to algorithmically solve the problem of what kind of feedback to seek, how much feedback the system needs to invite, and how to assimilate the feedback for maximum benefit. These problems can potentially modelled as probabilistic models that can be used to improve the decision making of the matching stage during bootstrapping.

Another possible future direction is handling unsatisfactory feedback provided by users. In recent work, Jeffery et al. [2008] stressed the fact that the assumption that user answers are accurate all the time does not always hold, as one cannot assume that the users always have knowledge of the domain or that they correctly understand the task in hand. The Bayesian inference methodology would be able to deal with uncertain situations with relation to the user feedback provided. For example, one would be able to judge whether a user is providing consistent answers by considering of asking users the same question and then updating the degree of belief in the answers by certain users, above other by confirming or disconfirming a hypothesis related to that.

In addition, further work could explore additional types of feedback that can be propagated to the schema inference methodology. For example, to change the cluster membership, or to select a different clustering solution by considering a different silhouette value. This could be beneficial since the schema inference methodology is based on the annotations from the clusters derived at the end of the cluster analysis. At this stage of the research, the exact manner as to how this can be achieved is not obvious.
5.2.2 Experimental Evaluation

In Chapter 4 we presented an experimental evaluation of the Bayesian updating approach in relation to matching. The evaluation was based on the judgements of human experts when judging construct equivalence in the presence of different kinds of syntactic and semantic evidence. The purpose of the experiment was to measure the effect on the quality of decisions made by string-based matchers during the matching stage of a dataspace life-cycle. To the best of our knowledge, there are no established benchmarks for judging individual aggregated degrees of belief or measuring uncertainty on decisions in matching. We therefore constructed a gold standard grounded of human experts.

Additionally, through our empirical observations on the WoD we noticed that the scarcity of cross-ontology links at the conceptual level, restricts us from making the most of the potential for using semantic evidence. In the future, we expect that the WoD will be more integrated with more links at the vocabulary level and across vocabularies thus enabling greater benefits from the assimilation of semantic knowledge (as construed in this dissertation). Our Bayesian methodology, would lead to increased certainty on the decisions made by string-based matchers. A possible future direction here is to explore which combinations of semantic evidence cause the most reduction on uncertainty, in which circumstances and scenarios.

5.2.3 Developed Techniques

Structure Inference for LD Sources

The evaluation of the structure inference technique (in Section 3.4) revealed that subsumption relationships can also be inferred. Doing so would further extend the expressiveness of the conceptual summaries suggested by the technique and it is a strong candidate for future improvement. Additionally, it would be interesting to observe the behaviour of the technique with different similarity measures than the Jaccard similarity over local names of predicates that we have used. We expect that the simplicity of the distance function in more complex RDF graph could potentially produce undesirable results. The exploration of an alternative similarity measure to the one currently used is a potential future improvement.
Managing uncertainty while matching LD sources

Our contributed framework for the assimilation of evidence considers various kinds of syntactic evidence. However, it only considers similarity scores from string-based matchers that work over local names from classes or properties defined as Web resources. Empirically modelling the likelihoods of more matchers on other sources of syntactic evidence (e.g., from the values of `rdfs:label` or `rdfs:comment`) would help reduce the uncertain decisions in relation to matching in dataspaces in a setting where the underlying sources are LD datasets even further. Regarding the semantic evidence used, our methodology considers explicitly defined semantic relations at the conceptual level. Reasoning on the conceptual information gathered would make more semantic evidence explicit thereby allowing it to be used for improving the decision making. Moreover, it would be interesting to derive probability distributions for other primitive matchers that derive a confidence score using ontologies or WordNet relationships as evidence.

Finally, the Bayesian framework, as it stands, makes judgements on a single hypothesis in relation to the matching stage of the life-cycle for deriving one-to-one semantic correspondences i.e., on construct equivalence. However, the framework has the potential to reason about different hypotheses, in different stages of the cycle, such as, on learning schema mappings or on learning degrees of belief for semantic correspondences as described by Guo et al. [2013]. The exploration of such questions could help in the construction of a comprehensive treatment of uncertainty across the entire life-cycle of a dataspace for LD.
Bibliography


tensible framework for high-performance dataset analytics. In EKAW, pages
353–362.


ontology matching with COMA++. In Proceedings of the ACM SIGMOD
International Conference on Management of Data, Baltimore, Maryland, USA,

trieval. ACM Press / Addison-Wesley.

W3C Team Submission. http://www.w3.org/TeamSubmission/turtle/; retr.
2014/05/03.

http://www.w3.org/TR/rdf-syntax-grammar/.

Belhajjame, K., Paton, N. W., Embury, S. M., Fernandes, A. A. A., and Hedeler,

Belhajjame, K., Paton, N. W., Fernandes, A. A. A., Hedeler, C., and Embury,
S. M. (2011). User feedback as a first class citizen in information integration
systems. In CIDR, pages 175–183.

uating schema matching and mapping. In Bellahsene, Z., Bonifati, A., and
Rahm, E., editors, Schema Matching and Mapping, Data-Centric Systems and

pages 451–460.


BIBLIOGRAPHY


BIBLIOGRAPHY


Appendix A

Density Estimation

A probability density function (a.k.a density function) represents, with a smooth curve, the probability distribution of a continuous random variable. The contribution described in Chapter 4 requires we estimate the univariate probability density functions for the continuous random variable associated with similarity scores returned by string-based matchers. This section proposes, for this particular purpose, the use of a non-parametric approach known as kernel density estimation (a.k.a kernel estimator).

A.1 Kernel Density Estimation

A widely used non-parametric approach for estimating the density function of a random variable $X$ is the kernel density estimation (KDE) [Silverman, 1986, pp.1–6]. As the approach is non-parametric it relies on a given set of independent observations $x_1, x_2, \ldots, x_n$ of $X$. According to Parzen et al. [1962], the univariate kernel density estimator is:

$$\hat{f}_X(x) = \frac{1}{nh} \sum_{i=1}^{n} K \left( \frac{x - x_i}{h} \right),$$

(A.1)

where $K(\cdot)$ is a weighting function, known as the kernel, that integrates to 1 and $h > 0$ is a smoothing parameter, known as the bandwidth. The shape of the weighting function is determined by the kernel’s standardised function, whereas the bandwidth controls the width of the kernel function and hence the amount of smoothing applied to $\hat{f}(x)$ [Silverman, 1986, pp.43-60]. Briefly, this is saying that the kernel density estimate of $f_X$ at the point $x$ is given by Equation A.1. It
is worth mentioning that each time a new density estimate is required, the entire series of observations has to be used.

A common choice [Sheather et al., 2004] for the kernel function is the Gaussian kernel, that is

\[ K(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}t^2\right). \] (A.2)

The choices of bandwidth, as well as of the kernel function\footnote{Examples of alternative kernels are Epanechnikov and Biweight the interested reader is referred to [Silverman, 1986, p.43] for more details.} are the topic of ongoing research. For a survey the reader is referred to [Jones et al., 1996]. Choosing large values for \( h \) causes \( \hat{f}(x) \) to over-smooth, and to under-smoothing otherwise. Determining the value of \( h \) and the choice of the kernel function that optimise the properties of \( \hat{f}(x) \) is beyond the scope of this dissertation. We used the routines implemented in MATLAB [MATLAB, 2013] to derive the value of \( h \) and perform density estimation using a Gaussian kernel.

The resulting KDE is used in the context of a framework to assimilating different kinds of evidence. In brief, Section 4.6.1 describes an experimental procedure used to observe the behaviour of string-based matchers in terms of the similarity scores they return when judging on the basis of syntactic similarity, the extent to which two constructs are equivalent. The observations derived in each case (equivalent/non-equivalent) are then used for the construction of a distribution function for similarity scores. Similarity scores computed by each matcher according to some syntactic metric (such as edit-distance) can be modelled as a continuous-valued random variable. In this case, the kernel estimator is used to approximate the p.d.f. of similarity scores. Having obtained the estimated p.d.f. \( \hat{f}(x) \), the function is used (as described in Section 4.6.3) to compute the probability that the random variable takes on a similarity score in a given interval. This process is a prerequisite for deriving the likelihood of a similarity score, which is, in turn a prerequisite for Bayesian inference.
A.2 Boundary Bias Reduction in Kernel Density Estimation

When the domain of values that a random variable can take is bounded on a specific domain e.g., [0, 1], direct application of the non-parametric approach for estimating the probability density function using a kernel estimator, such as in Equation A.2, is not desirable. This is because the estimator will suffer from well-known boundary bias problem [Silverman, 1986, p.29]. Such estimators allocate weight outside the bounded domain which causes boundary effects to arise. There are several well-known approaches for reducing such effects (see [Karunamuni and Alberts, 2005]). This issue needs to be taken into consideration when estimating the p.d.f.s for the similarity scores returned by each of the string-based matchers (for complete details on an experiment on how this can be done see Section 4.6).

A naive approach is to truncate the estimated density \( \hat{f}_X \) to the boundaries \([\alpha, \beta]\). However, this leads to biased estimates [Silverman, 1986, p.29]. In addition, the estimated p.d.f. derived by truncation no longer integrates to unity, and since the boundary conditions are ignored, it can lead to inaccurate estimates of the degrees of belief for the likelihoods required for performing Bayesian inference.

A simple method for estimating the p.d.f. with bounded support is to use the transformation method [Marron and Ruppert, 1994]. In simple terms, the transformation method estimates the p.d.f. of a transformed random variable \( Y = t(X) \). Assume a transformation function is given, such as \( t(X) = \log\left(\frac{x - l}{x - u}\right) \), with boundaries \([l, u]\) (e.g., \([0, 1]\)). The following three-step process is then carried out:

1. Transform the observation points \( x_1, x_2, \ldots, x_n \) of \( X \) using the log transformation function \( t(X) \) to obtain a new vector of observations \( y_1, y_2, \ldots, y_n \) of \( Y \) where \( y_i = t(x_i) \);

2. Apply the kernel estimator (as of Equation A.2) to estimate the p.d.f of \( t(x) \) using \( y_1, y_2, \ldots, y_n \) of \( Y \).

3. The result in (2) is then multiplied by the derivative of \( t(x) \), i.e., \( t'(x) = \frac{1}{x - l} + \frac{1}{x - u} \).

Suppose that the p.d.f. of \( Y \) is denoted by \( g(x) \), then \( \hat{f}(x) = g(t(x))t'(x) \) gives an estimate of \( f(x) \). In our contribution, this three-step process as described
in [Marron and Ruppert, 1994] is used to derive an estimate of the p.d.f. for the similarity scores returned by each string-based matcher, with bounded support in the domain [0, 1].
Appendix B

Mutation Procedure

Description of different mutations that a systematic procedure is introducing over local-names. We recognise that there are perhaps more mutations that can happen over the strings extracted from local-names. For our purposes the mutations described in Table B.1 are sufficient for deriving enough similarity score observations that are used as input to an approach for estimating the probability distributions for each string-based matcher.

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MISSED-CHR</td>
<td>Given a string of characters, choose at random a character to drop. Random choices follow a uniform distribution with all characters having equal probability of being selected.</td>
</tr>
<tr>
<td>TRANS-CHR</td>
<td>Given a string of characters, choose at random a character to transpose with the character on the immediate next position. Random choices follow a uniform distribution with all characters having equal probability of being selected.</td>
</tr>
<tr>
<td>DOUBLE-CHR</td>
<td>Given a string of characters, choose at random a character to duplicate. Random choices follow a uniform distribution with all characters having equal probability of being selected.</td>
</tr>
<tr>
<td>INSRT-CHR</td>
<td>Given a string of characters, choose at random a character to replace with a random character. Random choices follow a uniform distribution with all characters having equal probability of being selected.</td>
</tr>
<tr>
<td>TRANS-CHR-GRAMS</td>
<td>Given a string of characters, derive the tri-grams that exist, choose a tri-gram at random, transpose a character at random from the selected tri-gram. Random choices follow a uniform distribution with all characters having equal probability of being selected.</td>
</tr>
<tr>
<td>TRANS-ALL-CHRS</td>
<td>All characters are transposed at random.</td>
</tr>
</tbody>
</table>

Table B.1: Description of different types of mutations on local-names.
Given as input an ontology, the procedure returns a mutated version of that ontology along with the reference alignment that includes all the mappings between the original and the mutated version of the ontology. The algorithm follows a similar procedure to the test-cases generated by the OAEI, with the difference that it focuses on introducing mutations on the strings of local-names of constructs (which can be either classes or a properties). The procedure is configured with two parameters: *mutation rate*, and *mutation mode* that allow different run configurations. For instance, the procedure can be configured to introduce mutations with a different rate, this is controlled by the *mutation rate* parameter. If a *mild mutation* is selected as the value of the mutation rate parameter, there is a 20% chance for a mutation to occur over some construct and a 80% chance that a mutation will not occur. On the other hand, if the algorithm is configured with a *severe mutation* as the value of the mutation rate parameter, then there exists a 50% chance of a mutation to happen over the local-name of some construct. In addition to that, the *mutation mode* parameter decides on whether to generate mutations targeting specifically \textit{ed} or \textit{ng}. Table B.1 summarises the different types of mutations, whereas, Table B.2 shows different configurations of the algorithm with the type of mutations that can be introduced along with their probability of occurrence.

<table>
<thead>
<tr>
<th>Mutation Rate</th>
<th>Mutation Mode</th>
<th>Syntactic Mutations</th>
<th>Probability of occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mild</td>
<td>Edit-distance</td>
<td>MISSED-CHR</td>
<td>25%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INSRT-CHR</td>
<td>25%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANS-CHR</td>
<td>25%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DOUBLE-CHR</td>
<td>25%</td>
</tr>
<tr>
<td></td>
<td>N-gram</td>
<td>MISSED-CHR</td>
<td>25%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INSRT-CHR</td>
<td>25%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANS-CHR</td>
<td>25%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANS-CHR-GRAMS</td>
<td>25%</td>
</tr>
<tr>
<td>Severe</td>
<td>Edit-distance</td>
<td>MISSED-CHR</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INSRT-CHR</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANS-CHR</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DOUBLE-CHR</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANS-ALL-CHRS</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>N-gram</td>
<td>MISSED-CHR</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INSRT-CHR</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DOUBLE-CHR</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANS-CHR</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANS-CHR-GRAMS</td>
<td>20%</td>
</tr>
</tbody>
</table>

Table B.2: Probability of occurrence of mutations for different configurations.
Appendix C

Evidence Precedence Proof

Proof. First, we substitute Equation C.1 where the computed posterior, which considers a semantic evidence, assumes the role of the prior in the computation of the revised posterior, when the new evidence is a similarity score, computed using Equation C.2. To keep the notation clear, the subscript notation introduced in previous versions of the equations is omitted.

\[ p_{H|E}(h|e) = \frac{p_{E|H}(e|h)p_{H}(h)}{\sum_{i=1}^{n} p_{E|H_i}(e|h_i)p_{H}(h_i)} \]  \hspace{1cm} (C.1)

\[ p_{H|E}(h|e) = \frac{f_{E|H}(e|h)p_{H}(h)}{\sum_{h \in \text{Range}(H)} p_{H}(h)f_{E|H}(e|h)} \]  \hspace{1cm} (C.2)
\[ p(h|e) = \frac{p(e|h)p(h)}{p(e|h)p(h) + p(e\neg h)p(\neg h)} \]
\[ p(h|e) = \frac{f(e|h)p(h)}{f(e|h)p(h) + f(e\neg h)p(\neg h)} \]
\[ = \frac{f(e|h)p(e|h)p(h)}{p(e|h)p(h) + p(e\neg h)p(\neg h)} \times \frac{1}{f(e|h)p(h) + f(e\neg h)p(\neg h)} \]
\[ = \frac{f(e|h)p(e|h)p(h)}{p(e|h)p(h)(f(e|h)p(h) + f(e\neg h)p(\neg h)) + p(e\neg h)p(\neg h)(f(e|h)p(h) + f(e\neg h)p(\neg h))} \]
\[ = \frac{f(e|h)p(e|h)p(h)}{p(e|h)p(h)f(e) + p(e\neg h)p(\neg h)f(e)} \]
\[ = \frac{f(e|h)p(e|h)p(h)}{f(e)(p(e|h)p(h) + p(e\neg h)p(\neg h))} = \frac{f(e|h)p(e|h)p(h)}{f(e)p(e)} \]
Now, we substitute Equation C.2 where the computed posterior, which considers a similarity score, assumes the role of the prior in the computation of the revised posterior, when the new evidence is a semantic evidence, computed using Equation C.1.

\[
p(h|e) = \frac{f(e|h)p(h)}{f(e|h)p(h) + f(e\neg h)p(\neg h)}
\]

\[
p(h|e) = \frac{p(c|h)p(h)}{p(e|h)p(h) + p(e\neg h)p(\neg h)}
\]

\[
= \frac{f(e|h)p(e|h)p(h)}{f(e|h)p(e|h)p(h) + f(e|h)p(e\neg h)p(\neg h) + f(e\neg h)p(\neg h)p(e|h)p(h) + f(e\neg h)p(\neg h)p(e\neg h)p(\neg h)}
\]

\[
= \frac{f(e|h)p(e|h)p(h)}{p(e|h)p(h)(f(e|h)p(h) + f(e\neg h)p(\neg h)) + p(e\neg h)p(\neg h)(f(e|h)p(h) + f(e\neg h)p(\neg h))}
\]

\[
= \frac{f(e|h)p(e|h)p(h)}{p(e|h)p(h)f(e) + p(e\neg h)p(\neg h)f(e)}
\]

which completes the proof.  \(\square\)
# Appendix D

## Similarity Score Discretisation

### Data Example

<table>
<thead>
<tr>
<th>Sim. Score Bins</th>
<th>Bin Width</th>
<th>Midpoint Values (x-axis)</th>
<th>Frequency</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0.067)</td>
<td>0.067</td>
<td>0.033</td>
<td>3</td>
<td>0.066</td>
</tr>
<tr>
<td>(0.067, 0.133)</td>
<td>0.067</td>
<td>0.100</td>
<td>1</td>
<td>0.022</td>
</tr>
<tr>
<td>(0.133, 0.200)</td>
<td>0.067</td>
<td>0.167</td>
<td>9</td>
<td>0.199</td>
</tr>
<tr>
<td>(0.200, 0.267)</td>
<td>0.067</td>
<td>0.233</td>
<td>17</td>
<td>0.375</td>
</tr>
<tr>
<td>(0.267, 0.334)</td>
<td>0.067</td>
<td>0.300</td>
<td>16</td>
<td>0.353</td>
</tr>
<tr>
<td>(0.334, 0.400)</td>
<td>0.067</td>
<td>0.367</td>
<td>37</td>
<td>0.817</td>
</tr>
<tr>
<td>(0.400, 0.467)</td>
<td>0.067</td>
<td>0.433</td>
<td>15</td>
<td>0.331</td>
</tr>
<tr>
<td>(0.467, 0.534)</td>
<td>0.067</td>
<td>0.500</td>
<td>25</td>
<td>0.552</td>
</tr>
<tr>
<td>(0.534, 0.600)</td>
<td>0.067</td>
<td>0.567</td>
<td>49</td>
<td>1.082</td>
</tr>
<tr>
<td>(0.600, 0.667)</td>
<td>0.067</td>
<td>0.633</td>
<td>15</td>
<td>0.331</td>
</tr>
<tr>
<td>(0.667, 0.734)</td>
<td>0.067</td>
<td>0.700</td>
<td>51</td>
<td>1.126</td>
</tr>
<tr>
<td>(0.734, 0.800)</td>
<td>0.067</td>
<td>0.767</td>
<td>37</td>
<td>0.817</td>
</tr>
<tr>
<td>(0.800, 0.867)</td>
<td>0.067</td>
<td>0.833</td>
<td>29</td>
<td>0.640</td>
</tr>
<tr>
<td>(0.867, 0.934)</td>
<td>0.067</td>
<td>0.900</td>
<td>21</td>
<td>0.464</td>
</tr>
<tr>
<td>(0.934, 1.001)</td>
<td>0.067</td>
<td>0.967</td>
<td>354</td>
<td>7.816</td>
</tr>
</tbody>
</table>

Table D.1: Similarity score observations organised to produce a histogram for n-gram (equivalent case).
Appendix E

Density Estimate with Different Kernels

Figure E.1: Comparison of density estimates for similarity score observations using a standard Gaussian kernel with the transformation Gaussian kernel that has bounded support [0,1].
Appendix F

SPARQL Queries

Listing F.1: Example SPARQL query used for discovering equivalent classes.

```sparql
SELECT DISTINCT ?c1 ?c2
WHERE {
    {?c1 a rdfs:Class .}
    UNION {?c1 a owl:Class .}
    UNION {?c1 a skos:Concept .}

    ?c1 rdfs:subClassOf ?c2 .

    FILTER (!isBlank(?c2) && ?c1 != owl:Nothing && ?c2 != owl:Nothing
            && ?c1 != owl:Thing && ?c2 != owl:Thing)
}
```

Listing F.2: Example SPARQL query used for discovering disjoint classes.

```sparql
SELECT DISTINCT ?c1 ?c2
WHERE {
    {?c1 a rdfs:Class .}
    UNION {?c1 a owl:Class .}
    UNION {?c1 a skos:Concept .}


    FILTER (!isBlank(?c2) && ?c1 != owl:Nothing && ?c2 != owl:Nothing
            && ?c1 != owl:Thing && ?c2 != owl:Thing)
}
```
Appendix G

CURIE Prefixes

A list of the CURIE ¹ prefixes used throughout the dissertation are summarised in Table G.1:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:</td>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a></td>
</tr>
<tr>
<td>rdfs:</td>
<td><a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a></td>
</tr>
<tr>
<td>skos:</td>
<td><a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a></td>
</tr>
<tr>
<td>owl:</td>
<td><a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a></td>
</tr>
<tr>
<td>list:</td>
<td><a href="http://jena.hpl.hp.com/ARQ/list#">http://jena.hpl.hp.com/ARQ/list#</a></td>
</tr>
<tr>
<td>foaf:</td>
<td><a href="http://xmlns.com/foaf/0.1/">http://xmlns.com/foaf/0.1/</a></td>
</tr>
<tr>
<td>dbp:</td>
<td><a href="http://dbpedia.org/resource/">http://dbpedia.org/resource/</a></td>
</tr>
<tr>
<td>dbo:</td>
<td><a href="http://dbpedia.org/ontology/">http://dbpedia.org/ontology/</a></td>
</tr>
<tr>
<td>freebase:</td>
<td><a href="http://rdf.freebase.com/ns/">http://rdf.freebase.com/ns/</a></td>
</tr>
<tr>
<td>dbpprop:</td>
<td><a href="http://dbpedia.org/property/">http://dbpedia.org/property/</a></td>
</tr>
<tr>
<td>lgdo:</td>
<td><a href="http://linkedgeodata.org/ontology/">http://linkedgeodata.org/ontology/</a></td>
</tr>
<tr>
<td>schema:</td>
<td><a href="http://schema.org">http://schema.org</a></td>
</tr>
<tr>
<td>music:</td>
<td><a href="http://www.kanzaki.com/ns/music">http://www.kanzaki.com/ns/music</a></td>
</tr>
<tr>
<td>mga:</td>
<td><a href="http://dbtune.org/magnatune/artist/">http://dbtune.org/magnatune/artist/</a></td>
</tr>
<tr>
<td>mgt:</td>
<td><a href="http://dbtune.org/magnatune/track/">http://dbtune.org/magnatune/track/</a></td>
</tr>
<tr>
<td>event:</td>
<td><a href="http://purl.org/NET/c4dm/event.owl#">http://purl.org/NET/c4dm/event.owl#</a></td>
</tr>
<tr>
<td>time:</td>
<td><a href="http://www.w3.org/2006/time#">http://www.w3.org/2006/time#</a></td>
</tr>
<tr>
<td>tl:</td>
<td><a href="http://purl.org/NET/c4dm/timeline.owl#">http://purl.org/NET/c4dm/timeline.owl#</a></td>
</tr>
<tr>
<td>dc:</td>
<td><a href="http://purl.org/dc/elements/1.1/">http://purl.org/dc/elements/1.1/</a></td>
</tr>
<tr>
<td>eShop:</td>
<td><a href="http://www.wiwiss.fu-berlin.de/suhl/bizer/eShop/eShop#">http://www.wiwiss.fu-berlin.de/suhl/bizer/eShop/eShop#</a></td>
</tr>
<tr>
<td>iswc:</td>
<td><a href="http://annotation.semanticweb.org/iswc/iswc.daml#">http://annotation.semanticweb.org/iswc/iswc.daml#</a></td>
</tr>
<tr>
<td>mo:</td>
<td><a href="http://purl.org/ontology/mo/">http://purl.org/ontology/mo/</a></td>
</tr>
</tbody>
</table>

Table G.1: Mappings of prefixes used.