Wright State University

CORE Scholar

[Browse all Theses and Dissertations](#)

[Theses and Dissertations](#)

2012

Linked Open Data Alignment & Querying

Prateek Jain

Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

Repository Citation

Jain, Prateek, "Linked Open Data Alignment & Querying" (2012). *Browse all Theses and Dissertations*. 606.
https://corescholar.libraries.wright.edu/etd_all/606

This Dissertation is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.

Linked Open Data Alignment & Querying

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

By

PRATEEK JAIN
B. Tech., DA-IICT, India, 2006

2012
Wright State University

COPYRIGHT BY

Prateek Jain

2012

WRIGHT STATE UNIVERSITY
SCHOOL OF GRADUATE STUDIES

August 21, 2012

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Prateek Jain ENTITLED Linked Open Data Alignment & Querying BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Amit P. Sheth, Ph.D.
Dissertation Director

Arthur A. Goshtasby, Ph.D.
Director, Computer Science Ph.D. Program

Andrew T. Hsu, Ph.D.
Dean, School of Graduate Studies

Committee on
Final Examination

Amit P. Sheth , Ph.D.

Pascal Hitzler , Ph.D.

Krishnaprasad Thirunarayan , Ph.D.

Peter Z. Yeh , Ph.D.

Kunal Verma , Ph.D.

ABSTRACT

Jain, Prateek . Ph.D., Department of Computer Science & Engineering, Wright State University, 2012.
Linked Open Data Alignment & Querying.

The recent emergence of the Linked Data approach for publishing data represents a major step forward in realizing the original vision of a web that can "understand and satisfy the requests of people and machines to use the web content" i.e. the Semantic Web. This new approach has resulted in the Linked Open Data (LOD) Cloud, which includes more than 295 large datasets contributed by experts belonging to diverse communities such as geography, entertainment, and life sciences. However, the current interlinks between datasets in the LOD Cloud as we will illustrate are too shallow to realize much of the benefits promised. If this limitation is left unaddressed, then the LOD Cloud will merely be more data that suffers from the same kinds of problems, which plague the Web of Documents, and hence the vision of the Semantic Web will fall short.

This thesis presents a comprehensive solution to address the issue of alignment and relationship identification using a bootstrapping based approach. By alignment we mean the process of determining correspondences between classes and properties of ontologies. We identify subsumption, equivalence and part-of relationship between classes. The work identifies part-of relationship between instances. Between properties we will establish subsumption and equivalence relationship. By bootstrapping we mean the process of being able to utilize the information which is contained within the datasets for improving the data within them. The work showcases use of bootstrapping based methods to identify and create richer relationships between LOD datasets. The BLOOMS project (<http://wiki.knoesis.org/index.php/BLOOMS>) and the PLATO project, both built as part of this research, have provided evidence to the feasibility and the applicability of the solution.

Contents

1	Introduction	1
1.1	Goals of this Dissertation	2
1.2	Contributions	2
1.2.1	Conceptual Contributions	2
1.2.2	Artifacts	3
1.3	Chapter Overview	4
2	Semantic Web and State of the Art	7
2.1	Introduction	7
2.2	Ontology	8
2.2.1	Domain Specific Ontology	9
2.2.2	Upper Level Ontology	10
2.2.3	Basic Relationships present in Ontologies	10
2.3	RDF	12
2.4	SPARQL	13
2.4.1	SPARQL Query Types	14
3	Linked Data	16
3.1	Technology	17
3.2	Linked Open Data	18
3.3	Applications	19
3.4	Challenges	21
3.4.1	Absence of Schema Level Links	21
3.4.2	Lack of Conceptual Description of Datasets	22
3.4.3	Lack of expressivity	22
3.4.4	Difficulties with respect to querying	23
4	Ontology Alignment for Concepts on Linked Open Data	28
4.1	Ontology Matching	28
4.1.1	History	28
4.1.2	Techniques	29
4.1.2.1	Name Matching	29
4.1.2.2	Description Matching	29
4.1.2.3	Constraint-based Matching	30

4.1.2.4	Instance based Matching	30
4.1.3	Tools	30
4.2	BLOOMS Approach	32
4.3	Evaluation	36
4.3.1	Evaluation: Ontology Alignment Evaluation Initiative Oriented Track.	39
4.3.2	Evaluation: Ontology Alignment Evaluation Initiative Benchmark Track	41
4.4	Related Work	42
5	Contextual Ontology Alignment of LOD	44
5.1	Introduction	44
5.2	Knowledge Requirements	45
5.3	Approach	46
5.3.1	Construct BLOOMS+ Forest	47
5.3.2	Compute Class Similarity	47
5.3.3	Compute Contextual Similarity	48
5.3.4	Compute Overall Similarity	51
5.4	Evaluation	52
5.4.1	Data Set	52
5.4.2	Experimental Setup	53
5.4.3	Results and Discussion	55
5.5	Related Work	56
5.6	Conclusion	57
6	Partonomical Relationship Identification on Linked Open Data	59
6.1	Introduction	59
6.2	Winston’s Approach to Part-of Relationships—Ontologized	61
6.3	Approach	66
6.3.1	Candidate Generation	66
6.3.2	Hypothesis Generation	68
6.3.3	Hypothesis Testing	69
6.4	Evaluation	71
6.4.1	Intra-Dataset Instance-Level Partonomy Discovery	71
6.4.2	Inter-Dataset Instance-Level Partonomy Discovery	74
6.4.3	Assertion of schema level links	76
6.5	Related Work	77
7	Querying Partonomical Relationship on LOD cloud	79
7.1	Introduction	80
7.2	Background	80
7.3	Challenges	82
7.4	PARQ Approach	82
7.4.1	System Architecture	82
7.4.1.1	Mapping Repository	82
7.4.1.2	Transformation Rule Generator	83
7.4.1.3	Query Re-writer	84
7.4.2	Meta-level Transformation Rules	84
7.4.3	Algorithm	85

7.4.3.1	Explanation	86
7.5	Evaluation	88
7.5.1	Geonames Results and Discussion	90
7.5.2	Administrative Geography Ontology Results and Discussion	92
7.5.3	Summary of Results and Limitations	94
7.6	Related Work	94
7.7	Conclusion	96
8	LOQUS: Linked Open Data SPARQL Querying System	100
8.1	Introduction	100
8.2	Motivation	101
8.3	Our Approach	103
8.4	Evaluation	107
8.5	Related Work	111
8.6	Conclusion and Future Work	111
9	Conclusion	112
9.1	Summary	112
9.2	Further Work	114
9.2.1	Richer Relationship Identification on LOD	114
9.2.2	Yellow Pages for LOD	115
9.2.3	Flexible Question Answering using LOD	115
9.2.4	Property Matching on LOD	116
9.2.5	LOD Integration and Enhancement	117
9.3	Final Remarks	118
	Bibliography	120

List of Figures

2.1	Example of an ontology. Source: http://knoesis.org/research/semweb/projects/stt/	10
2.2	Example of an RDF Graph. Source: http://www.w3.org/TR/rdf-primer/	13
3.1	RDF Interlinking between different datasets using	18
3.2	Datasets available as part of LOD in May 2007	19
3.3	Datasets available as part of LOD in 2009	20
3.4	Possible LOD integration with SUMO	26
4.1	BLOOMS trees for Jazz Festival with sense Jazz Festival and for Event with sense Event. To save space, some categories are not expanded to level 4.	34
5.1	BLOOMS+ trees for Record Label and Music Company	49
6.1	PLATO System Architecture	67
7.1	PARQ system flow chart	97
7.2	PARQ Results on Geonames	98
7.3	Comparison PSPARQL and PARQ on Geonames for respondent 4	99
7.4	Comparison for Ordnance Survey Dataset for Respondent 4	99
8.1	LOQUS Architecture	104

List of Tables

3.1	Some Datasets that are Part of LOD Cloud	18
4.1	Results on the oriented matching track. Results for RiMOM and AROMA have been taken from the OAEI 2009 website. Legends: Prec=Precision, Rec=Recall, A-API=Alignment API, OMV=OMViaUO, NaN=division by zero, likely due to empty alignment.	39
4.2	Comparison of various systems on the benchmark track. Results for RiMOM and AROMA have been reused from the OAEI 2009 website. Legends: Prec=Precision, Rec=Recall	41
5.1	Common nodes between the two trees in Figure 5.3.2, and their depth. The first column gives the common nodes between the two trees rooted at <i>Record Label</i> and <i>Music Industry</i> . The second column gives the depth (the distance from root) of these nodes in the BLOOMS+ tree rooted at <i>Record Label</i> – i.e. the source tree.	50
5.2	Sample mappings of LOD ontologies to PROTON.	53
5.3	Results for various solutions on the task of aligning LOD schemas to PROTON. Legend: S-Match-M=Result of S-Match Minimal Set, S-Match-C=Result of S-Match Complete Set, Prec=Precision, Rec=Recall, F=F-Measure PRO=PROTON Ontology, FB=Freebase Ontology, DB=DBpedia Ontology, GEO=Geonames Ontology	55
5.4	Sample of correct mappings from LOD ontologies to PROTON generated by BLOOMS+ . .	55
5.5	Sample of incorrect mappings from LOD ontologies to PROTON generated by BLOOMS+ .	56
6.1	Six type of partonomic relation with relational elements	62
6.2	Precision of the six different relation types between DBpedia entities	73
6.3	This table shows PLATO’s performance on precision and recall for the Dish-Ingredient task, and PLATO’s performance on precision for the Anatomy-Organ task. Recall was not reported for the second task because of time and resource limitations.	75
6.4	Precision as measured on Schema Level Links Between DBpedia entities	76
7.1	Important Properties in Geonames	88
7.2	Important Properties in Administrative Geography Ontology	89
8.1	Result execution of queries over geonames	107
8.2	Result execution of queries over dbpedia	107
8.3	Result execution of queries over linkedmdb	108
8.4	Result of user submitted query	108
8.5	Result execution of queries using LOQUS	109
8.6	Comaparison LOD SPARQL Query Processing Systems	110

Acknowledgement

I would like to thank my advisor Amit P. Sheth for all his advice, guidance and support. I have been lucky enough to work with a brilliant professor. Professor Sheth has taught me how to have a long term vision and choose important problems to solve. He has taught me the importance of grounding my research. The common factor between the two of us is the desire to pursue research that will be valuable both in the short term and the long term. Over the years, I have learnt a lot from him and hope to maintain this relationship for a long time.

I want to express my gratitude to the members of my dissertation committee - Pascal Hitzler, Kunal Verma, Peter Z. Yeh and Krishnaprasad Thirunarayan. I truly enjoyed my interaction with them and appreciate the extremely valuable feedback they have provided to me over this time. I would like to thank Pascal, Peter and Kunal specifically. I have been very fortunate to have worked with them on several wonderful projects that have contributed immensely towards this dissertation. They are extremely talented and great researchers and wonderful collaborators as well. They are also true and wonderful friends. They taught me the art of identifying the correct problem and presenting solutions. Kunal and Peter took personal interest in my research and its application in industry. I am sincerely thankful to them for giving me the opportunity to be an intern under their guidance at Accenture Technology Labs.

I want to thank Cory Henson for a wonderful time in the graduate school. I have thoroughly enjoyed our discussions related to research, football, baseball and life in general. The many helpful comments and feedback I received from him have also been extremely valuable. He is a wonderful friend and I wish him my best for the career ahead.

I would like to express my appreciation to members (both past and present) of the Kno.e.sis Center. In particular I would like to thank Pavan Kapanipathi, Sarasi Lalithsena and Ajith Ranabahu for their help and co-operation during my studies. It was my pleasure to have known and interacted with these wonderful folks.

I want to acknowledge Tonya Davis, Valerie Smith, Jennifer Limoli, Paula Price and Wendy Chetcuti and

other wonderful members of the department's staff for always being ready to help. I want to especially thank Tonya. Tonya has been wonderful. She made my life a lot easier by handling the paperwork for my funding, reimbursing me for my travel and making sure that I got paid of time every month.

Another set of individuals who have helped me immensely to grow into a researcher is Dr. Sanjay Chaudhary, Dr. Vikram Sorathia and Dr. Zakir Laliwala. They were my mentors during my undergraduate degree program and provided me the initial push and direction. They believed in my capabilities and provided me with opportunities to grow and succeed. For this and everything else, I express my sincere gratitude to them.

I want to give a big thank you to my parents and siblings for their unconditional love, support and encouragement. They never questioned my decision to pursue a Ph.D. and they were always there when I needed support. Since my childhood they did everything in their power to help me achieve my career goals. I would not have made it this far without their support.

Last but not the least, I would like to thank Jennifer Prather for her unconditional love, support and encouragement. She has been wonderful throughout this journey. She made the hard times softer and good times that much better. She happily allowed me to rant during tough times and patiently gave me her valuable advice and suggestions. Thank you Jennifer for proof reading and correcting my papers, dissertation and job application emails. Her family members (including Tux and Radar) have equally contributed in making this journey easier and joyful.

The funding for this research work was provided by NSF as part of Award: IIS-0842129 titled *III-SGER: Spatio-Temporal-Thematic Queries of Semantic Web Data: a Study of Expressivity and Efficiency* and NSF Award 1143717 *III: EAGER – Expressive Scalable Querying over Linked Open Data*. These projects provided the framework within which the experiments described in this dissertation were conducted.

Dedicated to
My family and friends

Introduction

Digital data is omnipresent and is nowadays an integral part of our life. For the first time in the existence of humanity digital devices and sensors are capturing more snapshots and corresponding data than they can process and analyze. This data flood calls for new platforms for storage, analysis and computation which can keep up with the influx and make sense of this deluge. A popular term which technology analysts have used to describe this data revolution is 'Big Data'.

A recent report by McKinsey Global Institute [78] highlights some interesting statistics about big data. In 2010, more than 4 billion people or 60 percent of the world's population was using mobile phones. About 12 percent of these people used a smart phone and thus effectively acting as sensors and contributing in the growth of big data. The report goes on to report that more than 30 million sensor nodes are active in various sectors of economy and they are expected to grow at the rate of 30 percent annually. This never seen before flood of data has called for a change in the global computing paradigm. The report has a positive view about the change that can be unleashed because of the growth of data and its potential impact with some notes of caution and call for action. One such note is about changes required in technology and techniques in order to capture the full potential of the the data revolution. More specifically it talks about the need for innovation required to help individuals and organizations to integrate, analyze and consume the deluge of data. The report correctly warns that this data by itself is worthless and the holy grail lies in making sense of this data. This dissertation is a step in the direction of making sense of this data by achieving the goals of scalable data integration, querying and analysis.

1.1 Goals of this Dissertation

The goal of this dissertation is to develop a set of methodologies for systematically integrating, querying and analyzing big datasets. The objective is to develop the technique and also evaluate them on big data. More concretely the goals are as follows

1. Develop and evaluate techniques for data alignment which can cover heterogeneous and massive datasets related to various domains and contributed by community of users.
2. Develop and evaluate approach for scalable querying of datasets which can exploit the data alignment.
3. Develop and evaluate approach for richer relationships such as part-of between data entities.
4. Develop and evaluate approach for querying of datasets which can resolve the apparent mismatches between query constraints and data modeling.

1.2 Contributions

Conceptual contributions and the artifacts created to support the contributions are discussed next.

1.2.1 Conceptual Contributions

- The dissertation provides a conceptual framework for using crowd generated data to identify semantic relationships between entities. Most of the previous works in the field of ontology matching depend on using linguistics based or rule based techniques for the purpose of relationship identification. The techniques presented in this dissertation are unique as they rely on noisy data generated independently by people to alleviate the issues plaguing the LOD cloud. These techniques have been implemented by BLOOMS [61], BLOOMS+ [67] and PLATO systems.
- The dissertation presents the only approach of its kind which allows for identification of part-of relationships between entities. Further it also identifies the six different kind of part-of relationships

presented in [118]. This approach utilizes data from both within the structured web as well as unstructured web to identify this relationship. Identification of different types of relationships is extremely important in order to create new and intelligent applications.

- The dissertation presents an approach for querying LOD cloud or any other independently published data sources by creating an overarching schema. Using the relationships identified by the approaches in the previous two items, a schema can be generated for the purpose of reasoning and proper knowledge representation. This schema can be utilized for the purpose of query answering and processing without knowing the individual datasets. This approach makes it easier to query and identify relevant knowledge from the LOD datasets. The applicability and ease of this approach has been demonstrated using the LOQUS system [64].
- The applicability of these approaches has been validated empirically on big data datasets using real data from Linked Data cloud. The experiments demonstrate that these approaches are scalable and work well even in case of noisy data.

The work though is a small step in the overall idea of using information contained within the datasets to improve the data. The approach has a potential to be used for solving numerous other research challenges such as question answering and knowledge representation.

1.2.2 Artifacts

1. BLOOMS - An approach and system for bootstrapping ontology alignment using the LOD cloud. BLOOMS does not only significantly outperform state-of-the-art ontology alignment systems in LOD schema alignment; it also outperforms most other systems on the Ontology Alignment Initiative benchmark, and is roughly on par with the other best performing other system.
2. BLOOMS+ - An approach and system called BLOOMS+ for contextual ontology alignment of LOD ontologies. BLOOMS+ uses a more sophisticated metric to determine which classes between two ontologies to align and BLOOMS+ considers contextual information to further support (or reject) an alignment. A comprehensive evaluation of the solution using schema-level mappings from LOD ontologies to an upper level ontology is also presented. The results validate that the solution performed

well on this task and significantly outperformed existing ontology alignment solutions (including BLOOMS) on this same task.

3. PLATO - An approach and system for automatic detection of part-of relationships in the context of the LOD cloud. PLATO's approach of mining the Web to detect and validate the relationships for LOD cloud is rather unique and thus extends the existing arsenal of ontology engineering methods. The work also provides a formal representation of the partonomy classification created by Winston. The system has been evaluated to detect part-of relationships between hundreds of entities from prominent ontologies in the LOD cloud such as DBpedia and Freebase.
4. PARQ - An approach for rewriting SPARQL Queries, written from a users perspective without worrying about the underlying representation of information. The work utilizes partonomic transformation rules to re-write SPARQL queries. PARQ has been comprehensively evaluated on third party data (queries and dataset) and shows that it is able to re-write and answer queries not answered by a SPARQL processing system. We demonstrate PARQ can significantly improve precision without any recall loss.
5. LOQUS - An approach for querying LOD without knowing individual datasets. The system Linked Open Data SPARQL Querying System (LOQUS) allows users to effectively pose queries to the LOD cloud without having to know the exact structure and links between its many datasets. LOQUS automatically maps the users query to the relevant datasets (and concepts) using an upper level ontology; then executes the resulting query; and finally merges the results into a single, complete answer. A qualitative evaluation of LOQUS on several real-world queries demonstrates that LOQUS allows users to effectively execute queries over the LOD cloud without a deep understanding of its datasets. LOQUS is compared with existing query systems for the LOD cloud to highlight the pros and cons of each approach.

1.3 Chapter Overview

The dissertation is organized as follows: In Chapter 2 details about Semantic Web and the related technologies are presented. It discusses details about World Wide Web, its limitations and how Semantic Web

vision goes towards fixing some of the limitations. It gives details about ontologies which are the pillar of Semantic Web. It gives details about RDF and its usage in knowledge representation on the Semantic Web. The chapter concludes by giving details about the different kinds of relationships which occur in nature and are represented using ontologies.

Chapter 3 presents details about the principals for data interlinking on the Semantic Web known as the Linked Data. It describes the collection of interlinked ontologies which have been constructed using these principals. It identifies some of the applications created using these principals and datasets. The chapter concludes by identifying the challenges arising out of the interlinking of massive independent datasets.

Chapter 4 presents details about BLOOMS approach for Ontology Alignment on the Linked Data. It describes our technique which relies on bootstrapping and utilization of information from the web. The chapter also demonstrates a comprehensive evaluation of our approach on the publicly available datasets which are part of Linked Data cloud. We also demonstrate the effectiveness of this approach as a general purpose ontology alignment tool by showing results from publicly available benchmarks.

Chapter 5 builds upon chapter 4 in terms of enhancing the BLOOMS approach to take contextual information into account. It utilizes contextual information from and outside the ontologies to identify correct matches for concepts. The effectiveness of this approach is also demonstrated by showing results from publicly available benchmarks.

Chapter 6 presents the PLATO approach for identification of partonomical relationship between entities on the LOD cloud. The approach relies on using data within the ontologies and on the web to identify this relationship. The chapter also consists of a comprehensive evaluation showing the effectiveness of this approach on LOD ontologies.

Chapter 7 builds on Chapter 6 by presenting the PARQ approach for querying partonomical relationship from ontologies. The approach allows a user to query an ontology for partonomical relationship without knowing the granularity of modeling of information. The chapter presents a comprehensive evaluation of this approach with other state of the art systems using third party dataset and benchmarks.

Chapter 8 presents builds on Chapter 4,5 and 7 by presenting the LOQUS approach for querying information on the LOD cloud without knowing the individual datasets and relationships between them. The

LOQUS approach utilizes output from BLOOMS to identify the relationship between the individual datasets. The work presents an evaluation of this system with other state of the art systems for querying information from LOD cloud.

Finally, Chapter 9 concludes the dissertation with an overview of the research work, future directions where the research methodologies can be applied and ongoing work related to it.

Semantic Web and State of the Art

Semantic Web is widely understood to be a means of making content understandable to machines and agents using meta-data, reasoning and information integration [15, 14] and helping users find information easily. The basic idea of what 'Semantic Web' is, has been around for over two decades. Like World Wide Web it still relies on the use of URIs and using them to denote entities.

Slowly and steadily major industrial players have started realizing the potential and power of Semantic Web technologies. Consequently in the recent past there has been a big uptake in the adoption of Semantic Web technologies with major IT, pharmacy and bio-medical industries using the Semantic Web technologies for knowledge representation and discovery.

In this chapter, we cover the basic details related to Semantic Web and the various components which are involved in the making of semantic web.

2.1 Introduction

The idea of semantics or *meaning* is as old as the existence of humanity itself. Mankind has always been curious about the meaning of their existence, life and humanity. Right from its inception the focus has been on identifying relationships between entities, phrases, symbols and signs. Semantics as a field is related to various fields such as linguistics, formal logic and semiotics. It is perhaps quite apparent that the term Semantics is a phrase with multiple meanings and applicability. However, for the purpose of this dissertation usage of the term *semantics* is in regards to a web with meaning.

Much before Semantic Web, the concept of Semantic Network was introduced by Allan M. Collins,

Elizabeth F. Loftus and M. Ross Quillian in their research work ¹ in [26, 27]. The main idea behind their research work was to represent "semantically structured knowledge" by inserting metadata about hyperlinks interlinking web pages. Using this metadata intelligent agents could obtain more information and automate tasks such as information harvesting and performing simple tasks for users.

Sir Tim Berners-Lee who was also the inventor of the World Wide Web coined the term Semantic Web. He defines semantic web as "a web of data that can be processed directly and indirectly by machines" [13, 40]. After Sir Tim Berners-Lee became the director of World Wide Web Consortium (W3C), the idea of semantic web was placed under its umbrella. However, a lot of the technologies which play an important role towards fulfilling the vision of Semantic Web existed long before the adoption by W3C such as RDF, which originated from MCF ². Later on W3C published a specification for the RDF data model and XML based syntax in 1999. In 2004, a set of related specifications were released and most of the people are familiar with this version of RDF. Even before the advent of RDF, researchers in the AI community were making efforts to devise ways to capture and infer knowledge. AI researchers argued that by creating new computational models automated reasoning can be enabled. Around 1980, the AI community started referring to these models as 'ontologies'.

2.2 Ontology

'Ontology' as a term has different meanings depending on the perspective and the interest of the audience. The word 'Ontology' by itself means 'being', 'science', 'study'. It is an old phrase with roots in philosophy and logic. For the purpose of this dissertation though, the focus is on the term from a *semantic web* perspective.

In the area of Information Science, an Ontology formally represents the knowledge of a domain by describing (a) the various concepts related to the domain (b) the relationships between the concepts. Using an ontology it is possible to reason about the entities and draw new knowledge using the inferencing mechanisms. An ontology relies on a shared vocabulary and the vocabulary is used to model the domain.

¹http://en.wikipedia.org/wiki/Semantic_Web

²<http://www.guha.com/mcf/>

In [49] an ontology has been defined as "An ontology is a description (like a formal specification of a program) of the concepts and relationships that can formally exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set of concept definitions, but more general. And it is a different sense of the word than its use in philosophy."

An ontology has many different components which are used for capturing the various components of the domain. Some of these components are

1. **Individuals:** The objects or entities relevant to the domain which are to be represented. The notion of objects in ontology is similar to the notion of objects in Object-oriented Programming ³.
2. **Class:** An abstraction or generalization of the objects being represented. It can be also understood as a set or collection of the objects relevant to the domain.
3. **Attributes:** Attributes capture the value of the properties of the different entities. Both classes and instances can have attributes.
4. **Relations:** Relationships define ways in which instances and classes can be related to instances or classes.
5. **Restrictions:** Define conditions which must be true for the assertion to be valid. Restrictions are typically placed using properties.

Figure 2.1 shows an example of an ontology and the different components defined earlier.

Using these and other components a wide variety of ontologies can be constructed for the purpose of knowledge representation and reasoning. Ontologies are traditionally of two different types depending on the granularity of the modeling.

2.2.1 Domain Specific Ontology

: A domain ontology as the name suggests models a particular domain. It captures the various classes, entities and the relationships between them. An example of such as a domain specific ontology would be pizza ontology which models concepts and properties related to pizza.

³http://en.wikipedia.org/wiki/Object-oriented_programming

Figure 2.1: Example of an ontology. Source: <http://knoesis.org/research/semweb/projects/stt/>

2.2.2 Upper Level Ontology

: An upper level ontology represents objects and relationships in a way which is valid across various domain specific ontologies. For example, if there is a collection of ontologies about geography and events, an upper level ontology would abstract out the common concepts between them. These will include notions of space and time. An upper level ontology helps with resolving the mismatches which can arise between different domain ontologies. They usually contain a basic set of terms and relationships which can be applied across different ontologies. Example of some well known Upper Level Ontologies include SUMO [88], DOLCE [43] and PROTON [109, 31].

Ontologies are typically created using a W3C standard for creation of meta-model namely RDF.

2.2.3 Basic Relationships present in Ontologies

A key enabler of Semantic Web is the notion of relationships between entities [101] in an ontology. These relationships are asserted manually by the ontology creator or identified automatically between entities modeled in the ontology. While, it is possible to assert any relationship which occurs in nature in an ontology, there are certain core set of relationships which are very significant for knowledge representation using ontologies. Some of these core relationships have been described in [106, 118] and are as follows:

- is a : One of the most fundamental relationship which occurs in nature. It is used for indicating

an entity is of a specific type and possesses the attributes of that type. It is identical to indicating in Object-oriented programming that object x belongs to class y . Tools used for automatically identifying relationship between entities such as BLOOMS [61], BLOOMS+ [67] and others [23] predominantly identify *is a* relationship. An example of such relationship is *Joe is a person*.

- **part of:** This relationship is used to indicate relationship between an entity and other entity which are parts of it or make it. More details about this relationships are have been presented in a subsequent chapter of this dissertation and in [62]. An example of such relationship is *Dayton is located in Ohio*.
- **containment:** This relationship is present between an entity and another entity surrounding it. It is different from a part of relationship as the entities are not attached to each other but are only present within. An example of such relationship is *The president is in the White House*.
- **adjacency:** Adjacency relationship is very fundamental for any entity with spatial and temporal extent as it indicates other entities which lie at close proximity to it without any other entity being present in between. An example of such relationship is *Mexico is adjacent to United States*.
- **transformation:** It is a schema level relationship where members retain their identity but their classification is changed as a result of an operation. An example of such relationship is *an embryonic oenocyte changes into arval oenocyte. Thus preserving its identity but changing into a different kind of creature*.
- **derives from:** This relationship occurs between entities when an entity changes into another entity at a later point of time. This relationship is related to 'transformation' relationship but at the instance level. An example of such relationship is *coffee powder is derived from coffee beans*.
- **preceded by:** This relationship indicates a temporal relationship between two entities such as an entity or event occurred before the other entity or event. An example of such relationship is *snowfall is preceded by cold weather*.
- **succeeded by:** This relationship indicates a temporal relationship between two entities such as an entity or event occurred after the other entity or event. An example of such relationship is *sunrise is succeeded by dawn*.

An important objective of this dissertation is to build tools and techniques for automatic identification of some of these semantic relationships. However, before that can be achieved, we need ways to read and write the models which assert these relationships between entities. One such technique for reading and writing the semantic models is RDF.

2.3 RDF

Resource Description Framework or RDF is a method for modeling of web information and conceptual description. It is encoded using a number of ways such as Turtle [9], RDF/XML [8]. The main conceptual idea behind RDF is to make statements about resources especially Web Resources using the standard format of subject-predicate-object expressions. The subject denotes the entity or resource about which the statement is being made, whereas the Property indicates the information about the subject to be conveyed. The object captures the value of this property for the entity. For example a statement such as 'Wright State University is located in Dayton' can be denoted by making *Wright State* as the subject, *located in* as the property and *Dayton* as the object. RDF specification states that the subject should be a URI or a blank node. The property of any statement should be a URI. The object of a statement can be a resource or a literal. If the object of any statement is a URI, it can be reused as a subject for any other statement as well. Thus, a simple collection of these statements encoded using RDF constitutes a directed graph.

Figure 2.2 shows an example of an RDF graph.

In this graph simple statements such as resource denoted by <http://www.w3.org/People/EM/contact#me> has fullName Eric Miller is made. The same resource has email address em@w3.org.

A model used for representing knowledge is of limited use unless it can be queried. Just like for RDBMS, SQL is used for querying the underlying data, for RDF there is a querying language called SPARQL.

Figure 2.2: Example of an RDF Graph. Source: <http://www.w3.org/TR/rdf-primer/>

2.4 SPARQL

SPARQL is a recursive acronym for SPARQL Protocol and RDF Query Language. SPARQL is an RDF query language that can be used for querying and retrieving information stored using RDF format. Like any other typical query language SPARQL has a grammar and format and queries have to be expressed using this format.

A typical SPARQL query primarily consists of one or more triple patterns combined together using a clause for conjunction, disjunction or an optional pattern. Once a query is constructed it can be used to query local RDF data or a remote RDF store. It is also possible to distribute parts of a SPARQL query to different data sources and then integrating the answers after the individual queries have been answered.

The following is a sample of a SPARQL query on the RDF graph introduced earlier in figure 2.2.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
```

```
?person a foaf:Person.  
?person foaf:name ?name.  
?person foaf:mbox ?email.  
}
```

The simple query returns the name and email of any entity which is of type person in the RDF. *name* and *email* with the SELECT clause indicating the variables of interest. A variable in a SPARQL query is indicated by using a \$ and/or ? symbol. The remaining part following the WHERE clause denotes the triple pattern which should be matched with the underlying RDF graph for information to be returned by the SPARQL query.

2.4.1 SPARQL Query Types

SPARQL allows for four different query types which are as follows

1. SELECT Query : Used for obtaining values for variables which are indicating using a triple pattern or combinations of triple patterns from a datasource.
2. CONSTRUCT Query: Used for obtaining values from a graph and constructing a RDF graph using the values.
3. ASK Query: Used for getting a simple true or false answer from an RDF graph for a query.
4. DESCRIBE Query: Used for obtaining an RDF graph from the end point. The contents of this RDF graph and serialization depend on the implementation of the endpoint.

All the four different queries contain a WHERE clause to indicate the triple patterns of interest and the constraints.

Ontologies, RDF and SPARQL together provide the capabilities which are required to fulfill the original vision of semantic web [15]. However, all this data in isolation is of limited use. An ontology about tourist attractions is of limited use without data about places for food and lodging. An ontology about senator voting

records is of limited usage without data about industries in their congressional district. The semantic web community soon realized these limitations of the state of the art and started working interlinking of diverse but potentially related datasets by using RDF links [116]. This interlinking has lead to the emergence of *Linked Open Data*.

Linked Data

The term Linked Data refers to four simple principles specified by Sir Tim Berners-Lee for publishing and linking datasets to each other ¹. These four principles as specified on the webpage are as follows

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

While simple, these principles are powerful and have led to a revolution in data publishing on the web. Just like hyperlinks have helped in creating a web of documents, these principles have lead to the emergence of Web of Data [17]. The data in this collection consists of various domains such as life science, entertainment, government legislations and population.

In some cases the datasets are maintained by a single person such as 2000 U.S. Census in RDF ² while datasets such as Ordnance Survey Linked Data ³ are maintained by government organizations. It is believed that the LOD Cloud can significantly benefit both the AI and the Semantic Web communities by enabling new classes of applications and enhance existing tasks such as querying, reasoning, and knowledge discovery. To exemplify, *a scientist interested in exploring the relationship between the presence of the spider "Agelenopsis emertoni" and weather patterns*, can do so easily with the help of the LOD Cloud

¹<http://www.w3.org/DesignIssues/LinkedData.html>

²<http://thedatahub.org/dataset/2000-us-census-rdf>

³<http://thedatahub.org/dataset/ordnance-survey-linked-data>

as the Geospecies dataset gives information about the spider “*Agelenopsis emertoni*,” and the interlinking of Geospecies with Geonames makes it easy to explore the different kinds of information related to the locations where it can be found (*Wisconsin*), the locations where it cannot be found (*Iowa, Minnesota*), and the topography of these regions. Thus, in this scenario, the interlinks might help in programmatically identifying and analyzing the topographical patterns related to Iowa and Minnesota which make it difficult for this spider to survive in those regions.

3.1 Technology

Linked Data principles rely on using Uniform Resource Identifiers or URIs. URIs are very fundamental to the Semantic Web as all RDF Resources are supposed to be denoted using URIs [83]. While URLs are extremely popular and have been used as identifiers for Web documents, URIs provide a mechanism to identify any object which exists in the world but may not exist on the web [12]. They can also be used to denote abstract objects that are figments of our imagination. Since LOD heavily uses HTTP based technologies, it can be said to be built on top of the Web of Documents.

While documents on the Web are interlinked using hyperlinks⁴, RDF Links are in the form of RDF Triples [16]. The simple technique for RDF Links involves using a Subject from one namespace and/or dataset and an object belonging to a different dataset. Figure 3.1 shows a simple example of this interlinking. Here the Resource for the Major League Baseball team Cincinnati Reds in DBpedia is denoted to be same as Cincinnati Reds in Freebase. When the resource http://dbpedia.org/page/Cincinnati_Reds is looked up, the server returns an RDF graph providing more details about the team. This process of looking up the resource and getting the associated RDF graph back from the server for it is known as *dereferencing*. Similar kind of results are obtained when the lookup is done for the property `owl:sameAs` and the Freebase resource for Cincinnati Reds.

⁴<http://en.wikipedia.org/wiki/Hyperlink>

Figure 3.1: RDF Interlinking between different datasets using

3.2 Linked Open Data

The 4 linked data principles have been utilized by the Semantic Web community to link various ontologies to each other and create a massive collection of interlinked datasets known as the Linked Open Data ⁵. Table 3.1 lists some of the datasets available as a part of LOD Cloud.

Dataset	Description	Size in triples (approx)	Some datasets linked to
DBpedia	Information from Wikipedia	1 billion	Geonames, US Census, Freebase
Geonames	Geographic data	153 million	DBpedia, Jamendo, FOAF Profiles
US Census	2000 US Census data	1 billion	GovTrack, DBpedia, Geonames
GovTrack	Information about US Congress	2 billion	US Census
FOAFProfiles	Information about people	400k	SIOC, Flickr Exporter, Geonames

Table 3.1: Some Datasets that are Part of LOD Cloud

The initial push for publishing data using Linked Data principles came primarily from educational institutions as shown in Figure 3.2 and most of the data was related to entertainment, publication and geography. However, the community soon realized the advantage and potential in this publication paradigm and as shown in Figure 3.3, in a matter of couple of years, LOD moved onto to over 100 datasets with universities, private companies, government organizations all heavily involved in this process.

This growth and participation of various organizations continues to this day, and as of now, the LOD cloud consists of over 300 datasets from diverse domains such as life science, clinical trials, geography, legislations and scientific publications. This trend of publishing and linking data to other datasets is supposed to continue and contribute towards development of applications for knowledge acquisitions and discovery.

⁵<http://linkeddata.org/>

Figure 3.2: Datasets available as part of LOD in May 2007

The next section presents a short summary of such end user applications along with their brief description. Please note, by no means this list is comprehensive and it only includes end user applications.

3.3 Applications

BBC Music BBC Music is an online application which provides information related to music and entertainment. It provides a unique URI for the entities such as artists, sound tracks, concerts and venues ⁶. It consumes data from MusicBrainz ⁷ and Wikipedia (DBpedia), thus publishing data as well as consuming data from other sources. For example, it pulls in biographical information related to any artist from Wikipedia. Information about new music release related to any artist is pulled in from MusicBrainz.

Watson IBM Watson [42] is a Question and Answering system capable of parsing, understanding and answering queries provided in natural language. IBM Watson utilized some pieces of information available as a part of DBpedia dataset. The primary use of this information was to type candidate answers using type coercion [86]. Though it utilized a small chunk of the LOD datasets, its overall impact was around 5% towards improving the accuracy of the answers generated by Watson.

⁶<http://www.bbc.co.uk/music>

⁷<http://musicbrainz.org/>

Faviki Faviki is a social bookmarking site which allows people to use Wikipedia articles as tags for describing their entities of interest ⁸. The site relies on consuming DBpedia behind the scene and thus is a consumer of LOD dataset.

British Museum British Museum web portal provides information about various artifacts available in the various British Museums and information related to them [48]. The website consumes data from different LOD datasets such as DBpedia and provides more information to readers about these artifacts than would be normally available.

⁹<http://www.ibm.com/developerworks/webservices/library/ws-soa-linkeddata/index.html>

3.4 Challenges

The applications presented in the previous section clearly indicate that LOD datasets have been utilized by researchers and practitioners alike to extract knowledge and to discover new knowledge. However, the number of such applications is small and most of them utilize at most two dataset. There are practically no applications which utilize more than two datasets.

A glimpse at the current state of the interlinks and datasets indicate an important reason for the current state is due to the process used for knowledge modeling and interlinking. The current interlinks between datasets in the LOD Cloud are too shallow to realize much of the benefits promised. If this limitation is left unaddressed, then the LOD Cloud will merely be more data that suffers from the same kinds of problems which plague the Web of Documents, and hence the vision of the Semantic Web will fall short.

The growing number of datasets available on the LOD Cloud presents a challenge with regards to its usage, since on the one hand datasets such as DBpedia and Freebase offer massive amounts of information from diverse domains, while on the other hand there is no formal description of these or any other LOD Cloud components or their interlinking. If these issues can be alleviated then the LOD Cloud can be transformed from "merely more data" to "semantically linked data" by addressing the shortcomings identified in the following.

3.4.1 Absence of Schema Level Links

The LOD Cloud datasets lack schema level mappings and do not convey relationships between concepts of different datasets at the schema level. To exemplify, a feature in the Geonames schema can serve as a venue for an event, e.g the current model identifies *"Atlanta in Georgia was the venue of 1996 Olympics"* at the instance level. This creates significant limitations with respect to the reasoning potential which knowledge on the schema level would provide.

3.4.2 Lack of Conceptual Description of Datasets

Identification of the domain of a dataset requires manual intervention. For example, currently *there is no mechanism to describe that Jamendo¹⁰ captures music related information, whereas Geonames captures geographical information.* This is a serious drawback if we envision applications that could seamlessly harness the vast number of facts present in the cloud. Although some efforts have been made to devise a solution to describe the datasets [96, 1], these approaches focus more on the statistical aspects of the datasets and do not cater to the requirements for capturing conceptual information. The presence of a conceptual description will help in making knowledge discovery automatic and systematic.

3.4.3 Lack of expressivity

The LOD Cloud is of very shallow expressivity as a knowledge base and thus hardly allows to make use of underlying formal semantics through reasoning. The LOD Cloud primarily consists of ground level RDF triples, and hence does not utilize rich expressive features provided by OWL or RDF Schema. To exemplify, *there is inconsistency related to the population of Barcelona between DBpedia and Geonames. This could be detected (and hence fixed) by declaring the properties dbpedia-owl:populationTotal and geonames:population to be functional.* Since instances of Barcelona in geonames and DBpedia are linked to each other using `owl:sameAs`, using an OWL reasoner, an inconsistency could be detected, since an instance cannot have multiple values for a functional property. The lack of such expressive features is a severe drawback as expressivity enhanced LOD Cloud could significantly help in knowledge discovery and thus promote the usage of the LOD Cloud in the scientific community and elsewhere.

This brings to fore the need for conflict resolution techniques using additional contextual information about the sources/provenance. Otherwise, available information in sources such as DBPedia can never be used because of presence of conflicting information.

The shortcomings identified above severely impact the usage and limit the applications that can be built using the LOD Cloud. To justify our arguments, the following section illustrates the impact of these

¹⁰<http://dbtune.org/jamendo/>

shortcomings on an important requirement related to knowledge discovery, namely the seamless querying of the LOD Cloud.

3.4.4 Difficulties with respect to querying

SPARQL [100] has emerged as the de-facto query language for the Semantic Web community. It provides a mechanism with which a user can express constraints and facts, and the entities matching those constraints are returned to the user. To ease this process from an infrastructural perspective, data contributors have provided public SPARQL endpoints to query the LOD Cloud datasets. However, the syntax of SPARQL requires users to specify the precise details of the structure of the graph being queried in the triple pattern. To illustrate, in order to formulate a query which spans multiple datasets such as *"Select artists within Jamendo who made at least one album tagged as 'punk' by a Jamendo user, sorted by the number of inhabitants of the places where they are based"*, the user has to be familiar with multiple datasets, and has to express the precise relationships between concepts in the RDF triple pattern, which even in trivial scenarios implies browsing at least two to three datasets. In our previous work [65] we made progress towards alleviating this obstacle. But with respect to a systematic querying of the LOD Cloud we believe that the following challenges make the process difficult and will have to be addressed.

- *Schema heterogeneity*: The LOD Cloud datasets cater to different domains, and hence have been modeled differently. To exemplify, *a user interested in music related information has to skim through at least three different datasets such as Jamendo, MusicBrainz, MySpace*. This is perfectly fine from a knowledge engineering perspective, but it makes the querying of the cloud difficult as it requires users to understand the various heterogeneous schemas. This stems from the ***Lack of Conceptual Description of the Datasets*** as pointed out above. These issues are both at conceptual heterogeneity and syntactic heterogeneity. First requires non-trivial mapping between models, the second requires use of same-as or unit conversions between different syntax.
- *Entity disambiguation*: Often LOD datasets have overlapping domains and hence provide information about the same entity. To exemplify, both DBpedia and Geonames have information about the city of Barcelona. Although DBpedia references Geonames using the `owl:sameAs` property, from the

perspective of querying this makes it difficult as it might confuse the user as to which is the best source to answer the query. This problem gets even more compounded when contradictory facts are reported for the same entity by different datasets. *For example, DBpedia quotes the population of Barcelona as 1,615,908, whereas according to Geonames it is 1,581,595.* One can argue this might be because of difference in the notion of *the city of Barcelona*. But that leads to another interesting question: *Is the `owl:sameAs` property misused in the LOD Cloud?* This issue is partly related to **Lack of expressivity** since there is no mechanism to perform verification of facts. Additionally, the LOD methodology prohibits reification of statements, thus disallowing assignment of context to statements.¹¹ Researchers have recognized the severity of this issue and techniques for fixing this issue have been proposed in [19, 117]. But it is not clear, how these works can be directly applied in the problems highlighted above with respect to LOD Cloud.

- *Ranking of results:* In scenarios where the results of a query can be computed and returned by multiple datasets, the result which should be ranked higher for a specific query becomes an interesting and important question. As presented above, the query related to *population of Barcelona* can be answered by multiple datasets such as Geonames and DBpedia, but which one of them is more relevant in a specific scenario is a relevant question. This issue has been addressed from the perspective of popularity of datasets by considering the cardinality and types of the relationships in [110], but not from the perspective of requirements with regard to a specific query.

Some of the LOD Cloud shortcomings identified above can be resolved by providing a systematic and formal description of the LOD Cloud. There is an apparent lack of an ontology which formalizes and systematically captures the information contained in LOD Cloud datasets. Such an ontology would bring multiple benefits with respect to the use of the LOD Cloud by providing systematic descriptions of the domains captured by the datasets, schema level linking of the datasets, additional schema-level axioms, and hence also better reasoning capabilities. Typically, such an integration would make use of an upper level ontology.

Indeed, in the past the Semantic Web community has relied on upper level ontologies such as Cyc [99], SUMO [88], or DOLCE [81] to integrate heterogeneous knowledge bases. For applications, these ontologies have been integrated with domain specific ontologies [34, 89] to provide advantages such as

¹¹Note that even OWL, in the forthcoming revision OWL 2 [58], allows for some simple metamodelling.

better knowledge discovery, reasoning, or consistency verification.

An upper level ontology typically describes the knowledge base at a very abstract level and thus may or may not convey schema-level knowledge for the grounded knowledge bases which are part of the LOD Cloud. The presence of diverse datasets indeed calls for an ontology which is sufficiently abstract to be able to link to the diverse LOD datasets, but at the same time is grounded enough to provide for easy mapping to LOD datasets. For transforming the LOD Cloud from "merely more data" to "semantically linked data" this integration should provide the following features:

3.4.4.0.1 Systematic and Formal Description of LOD Datasets An upper level ontology captures various domains at a fairly abstract level. However the LOD extension of this upper level ontology should create a bridge between the abstraction of the ontology and instantiations available in the LOD Cloud. This will help in providing systematic and formal descriptions of the various ground statements, the classes to which the instances belong, and for identifying schema level relationships. As such, it will go a long way in creating a semantic description of the cloud, and thus help in identifying relationships between datasets at the schema level, and hence facilitate applications which need to perform reasoning over the cloud. Figure 3.4 depicts conceptually such an integration of SUMO with the LOD cloud. This issue has been recognized by other researchers and recently efforts have been made to utilize another well known upper level ontology Cyc [99] to provide a structural backbone to the LOD Cloud though UMBEL [10]. UMBEL contains schema level links to 21 different LOD Datasets, and thus is a much needed step in this direction. Another noticeable effort in this direction is the emergence of Linked Data Semantic Repository¹², which presents a reasonable view grouping of the several of the central datasets of the Linking Open Data (LOD) Cloud.

In a nutshell, at this time, there is no standard way of describing what an LOD dataset contains. Hence, there is a need for some kind of a conceptual description such as an ontology to describe these individual datasets.

3.4.4.0.2 Ease of Querying An integrated upper level ontology will help for querying since the specific branches of the upper level ontology will be linked to the LOD Cloud, hence the user knows which

¹²<http://ldsr.ontotext.com/>

Figure 3.4: Possible LOD integration with SUMO

sections of the cloud to look for. It also leaves scope for automated mechanisms for propagating queries over the cloud. To exemplify, if a user specified a SPARQL query in terms of the concepts of the upper level ontology, the mechanism will allow the query to propagate down and query data from actual datasets.

3.4.4.0.3 Checking Inconsistencies in the LOD Cloud An upper level ontology with axioms can help in detecting inconsistencies plaguing the linked data cloud. This extension can help in verification of the information captured by the LOD Cloud and thus identify and filter any inconsistent data. Inconsistencies, such as population of London¹³ can then be removed using this approach.

3.4.4.0.4 Ease of Maintenance and Extensibility Since the LOD Cloud continues to increase in size and will capture more diverse domains in the future, the extension should be easy to maintain to allow modifications, and should support extensibility to provide support for concepts which are not supported natively by the ontology.

¹³<http://iandavis.com/blog/2009/08/time-in-rdf-1>

Finally a note on scalability issues: While it could be argued, that an attempt to enhance the LOD Cloud with more expressive schema-level knowledge might be doomed from the start due to difficulty of dealing with very large amounts of schema knowledge in ontology reasoners, we believe that this is not necessarily the case. Recent advances, in particular those reported around the Billion Triple Challenges at the International Semantic Web Conferences,¹⁴ show that reasoning over very large knowledge bases is within reach. Importing such reasoning into realistic applications over realistic datasets, as those in the LOD Cloud, however, requires further advances into reasoning with large volumes of noisy data, and indeed research efforts need to be undertaken to realize this. A general discussion of the issues involved in this can be found in [57].

In the next chapter this work presents a solution to the problems outlined in **Absence of Schema Level Links**. The solution relies on utilizing the knowledge available on the Web and within the different LOD datasets to alleviate the lack of schema knowledge.

¹⁴<http://challenge.semanticweb.org/>

Ontology Alignment for Concepts on Linked Open Data

While LOD datasets are well interlinked on the instance level, they are very loosely connected on the schema level (see also Table 3.1). However, for tasks such as federated querying, knowledge discovery and reasoning there is a need for schema level relationships. One of the ways to identify schema level relationships is to investigate the field of Ontology Matching and experiment with state of the art ontology matching tools.

4.1 Ontology Matching

4.1.1 History

Identifying relationships between entities is a natural behavior of human mind. Humans have been doing it since our existence on this planet. However, the first comprehensive documentation about a taxonomic classification of relationships is Aristotles Organon [4]. The work introduces Aristotle’s classification of the different entities in along 10 different dimensions such as time, space, quantity and quality. While classification and identification of relationship has been covered in linguistics for eons, for the purpose of this dissertation the focus is on automated approaches to ontology matching.

Relational Databases [25] have long been used for the purpose of data modeling, storage and querying. Therefore, it is no surprise that the earliest work in this area is from the database community for schema matching. In [7] the authors have presented a comprehensive survey of some of the earliest techniques developed in the database community for the purpose of automatic schema matching. The work discusses a five

step methodology for schema integration which was followed by other researchers in subsequent surveys. In [76] the authors have discussed the concept of semantic and dynamic attribute capturing for four different kinds of mapping. These mappings involved syntactic, table, functional and program based mappings. In [102] the authors have presented one of the first automated tools for database administrators and developers to achieve schema mapping. In [97] the authors have presented one of the most comprehensive surveys of the different techniques utilized for the purpose of database and xml schema mapping. They classify the techniques into schema level matchers and instance based matchers. For both these classifications they classify these techniques by linguistics based, rule based and constraint based. The work also describes hybrid matchers which combine one of the more specific techniques used for the purpose of schema matching.

4.1.2 Techniques

Researchers have utilized various techniques for the purpose of schema matching. These approaches have been broadly classified by [97] as following:

4.1.2.1 Name Matching

This technique relies on identifies concepts and/or properties with similar names in the schemas. The similarity could be of different types such as (i) string similarity (ii) canonical name similarity (iii) synonym equality (iv) hypernym equality (v) similarity based on substrings, pronunciation and edit distances. (vi) name similarity provided by user.

Using this kind of similarity matching it is possible to identify more than 1:1 match. Hence, it is possible to match "phone" to both "home phone" as well as "office phone".

4.1.2.2 Description Matching

Schema elements utilize string comments in order to explicitly indicate the semantics of the given elements. This technique relies on finding the similarity between the string description to identify the relationship

between the elements. This technique could be as simple as finding the keywords from the string description and using them to match the elements. A more sophisticated methodology could involve parsing and understanding of the natural language description.

4.1.2.3 Constraint-based Matching

Constraints are often utilized in schemas to define the type of values which can be assigned to properties or classes. If both the schemas contain this kind of information, then it can be utilized by matchers to match elements with similar kind of constraints. This kind of approach is especially useful for owl based ontologies and/or database schemas. Both schemas utilize constraints to assign cardinality restricts, links to other entities and types of values that can be assigned. Thus by comparing the constraints assigned to any element it can be identified with which element is has the maximum semantic similarity.

4.1.2.4 Instance based Matching

Instances assigned to classes can be matched using one of the techniques mentioned earlier. Especially, linguistic based matchers , name and description matching are very useful for the purpose of this matching. Depending on relationships identified between instances, inferences can be made about schema level relationships between the classes of these instances.

Researchers have implemented and utilized these techniques to create state of the art systems for the purpose of schema matching. In the next section, we report on some of these tools.

4.1.3 Tools

In this section a report on some of the state of the art tools related to the area of ontology matching has been presented. These tools were selected due to the various techniques described in the previous section, used by them for the purpose.

- **RiMOM:** RiMOM is an ontology matching system which utilizes multiple strategies for performing the tasks [75]. The work identifies the similarity characteristics between different ontologies and em-

employs the technique which is best suited depending on the measure of similarity. The work considers both the textual and semantic similarities of the ontologies. The task of alignment is performed by using machine learning based techniques.

- **ASMOV:** ASMOV utilizes lexical and structural similarity between ontologies to calculate similarity measure between ontologies [68]. ASMOV combines these similarities with formal semantics in order to measure if there are any inconsistencies and identifies and fixes them. The system eventually computes a similarity matrix indicating the semantic similarity scores.
- **AROMA:** AROMA is an ontology matching system which uses hybrid and extensional matching methods to identify relationships between OWL ontologies. The system is capable of finding equivalence and subsumption relations between entities issued of OWL ontologies [33].
- **TaxoMAP:** TaxoMAP is a system which performs alignments between OWL ontologies. The system utilizes the label of the concepts and using syntactic analysis tool TreeTagger [79] generates similarity between these labels.
- **OMViaUO:** OMTViaUO utilizes upper level ontologies such as SUMO and DOLCE as semantic bridges in the ontology matching process [80]. The work demonstrates that the "nonstructural matching method" via OpenCyc and SUMO-OWL improves the precision and maintains the recall at the same time.
- **S-Match:** S-Match performs ontology matching between ontologies, taxonomies and catalogues. There are three different matching algorithms which are available with S-Match namely basic matcher, minimal matcher and structure preserving matcher.

However, it turns out that the performance of these systems on LOD schema datasets is rather poor, even though they performed fine on established benchmarks (see evaluation) . Thus, there was a need to find a unique solution to LOD schema alignment which is known as BLOOMS. Details about BLOOMS are reported in the next section.

4.2 BLOOMS Approach

BLOOMS relies on the utilization of a bootstrapping based approach. The system computes alignments with the help of noisy community-generated data available on the Web. Currently, BLOOMS uses Wikipedia and the Wikipedia category hierarchy for this purpose. However there is no conceptual reason why one would not be able to use other inputs (or even existing upper-level ontologies or upper-level domain-specific ontologies) instead. This would simply result in a different bias for the alignment, which could potentially be exploited, e.g., for alignment tasks on narrower thematic domains (see also discussion of future work, Section ??). Furthermore BLOOMS utilizes the Alignment API [36] as a base system by exploiting its capabilities which complement the native BLOOMS bootstrapping approach.

BLOOMS is a system for schema alignment. For the purpose of this dissertation, schema alignment means the generation of links between class hierarchies (taxonomies), which are `rdfs:subClassOf` relations. For an example, if "Human" occurs in some dataset and "Woman" occurs in some other dataset, then BLOOMS would be expected (or any other ontology alignment system) to create a relation between these two classes in the form of an RDF triple "Woman `rdfs:subClassOf` Human". Note that two classes A and B will always be related by one out of four relationships: A `rdfs:subClassOf` B , B `rdfs:subClassOf` A , A `owl:equivalentClass`¹ B , or none of the previous three.

At the core of the BLOOMS bootstrapping approach is the utilization of the Wikipedia category hierarchy. In essence, BLOOMS constructs a forest (i.e., a set of trees) T_C (which are known as the *BLOOMS forest* for C) for each matching candidate class name C , which roughly corresponds to a selection of super-categories of the class name. Comparison of the forests T_C and T_B for matching candidate classes C and B then yields a decision whether or not (and with which of the candidate relations) C and B should be aligned. It is spelled out in detail in the next section.

BLOOMS accepts as input two ontologies which are assumed to contain schema information. It then proceeds with the following steps.

1. **Pre-processing of the input ontologies** in order to (i) remove property restrictions, individuals, and

¹This is semantically equivalent to stating *both* A `rdfs:subClassOf` B and B `rdfs:subClassOf` A , and it is abstracted from the (syntactic) difference.

properties, and to (ii) tokenize composite class names to obtain a list of all simple words contained within them, with stop words removed.

2. **Construction of the BLOOMS forest** T_C for each class name C , using information from Wikipedia.
3. **Comparison of constructed BLOOMS forests**, which yields decisions which class names are to be aligned.
4. **Post-processing** of the results with the help of the Alignment API and a reasoner.

More details and examples on the key steps just described are now presented. As a running example, the class names Event and JazzFestival taken from the LOD datasets DBpedia and Music Ontology are used, respectively.

Pre-processing of the input ontologies. This involves a straightforward algorithm which normalizes each input class name C into a string C' obtained by replacing underscores and hyphens² by spaces, splitting at capital letters, and the like.³ For stop word removal the 319 stop words defined by the Information Retrieval Research Group of Glasgow University were used.⁴

For the running example, JazzFestival is transformed to "Jazz Festival", whereas Event is not modified at all.

Construction of the BLOOMS forest T_C from C . The first step in constructing T_C is to invoke a call to the Wikipedia Web service using C' as input. This Web service returns a set of Wikipedia pages⁵ W_C as results of a search on Wikipedia for the words in the string. If a returned result is a Wikipedia disambiguation page, it is then removed from W_C and replaced by all Wikipedia pages mentioned in the disambiguation page. The elements of the resulting set are called W_C *senses* for C .

Concerning the running example, for Event, the Web service returns Event, Eventing, Sport, NFL Draft, News, Festival, Event-driven programming, Rodeo, Athletics at the Summer Olympics, and Extinction event.

²The hyphens were removed manually, because they occurred only in one of the test ontologies, namely the AKT Portal Ontology (see Section 6.4).

³There was no need to make use of a dictionary, mainly because the resulting strings are used as input to Wikipedia search, which works well without stemming etc.

⁴http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words

⁵More precisely, their URLs.

Figure 4.1: BLOOMS trees for Jazz Festival with sense Jazz Festival and for Event with sense Event. To save space, some categories are not expanded to level 4.

In the next step, for each sense $s \in W_C$, a tree $T_s \in T_C$, called the *BLOOMS tree for C with sense s* , is constructed, as follows.

- The root of the tree is s .
- Children of s are exactly all the Wikipedia categories into which the Wikipedia page s is categorized.
- Subsequently, for each category c which is a node in the tree, its children are exactly all Wikipedia categories of which c is a subcategory.
- T_s is the resulting tree, which is cut at level 4 (i.e., branches of T_s have maximally 5 nodes, including the root).

The tree were cut at level 4 because the deeper levels involve Wikipedia categories which are very general, like "Humanities". These categories would be ineffective for the purposes.

Figure 4.1 shows the BLOOMS tree for Event with sense Event and for Jazz Festival with sense Jazz Festival.

Comparison of constructed BLOOMS forests. Any concept name C in the one input ontology is now matched against any concept name D in the other input ontology. This is done by comparing each $T_s \in T_C$

with each $T_t \in T_D$. For this, a function o is defined, which assigns a real number in the unit interval to each (ordered) pair of BLOOMS trees. The value $o(T_s, T_t)$, called the *overlap of T_t with T_s* , is defined as follows.

1. Remove from T_s all nodes for which there is a parent node which occurs in T_t . All leaves of the resulting tree T'_s are either of level 4 or occur in T_t . Note that due to the way BLOOMS trees are constructed, only nodes from T_s were removed which actually occur in T_t —They were removed because they do not give any essential additional information for comparing T_s with T_t .
2. $o(T_s, T_t) = \frac{n}{k-1}$, where n is the number of nodes in T'_s which occur also in T_t , and k is the total number of nodes in T'_s (The root is not counted).

In the running example, the BLOOMS trees in Figure 4.1 are pruned beneath the dark gray nodes. This results in $o(T_{\text{Event}}, T_{\text{Jazz Festival}}) = \frac{3}{4}$ and $o(T_{\text{Jazz Festival}}, T_{\text{Event}}) = \frac{3}{5}$.

The decision on an alignment is then made as follows.

- If, for any choice of $T_s \in T_C$ and $T_t \in T_D$, it implies that $T_s = T_t$, then C is set to owl:equivalentClass D .
- If $\min\{o(T_s, T_t), o(T_t, T_s)\} \geq x$ for any choice of $T_s \in T_C$ and $T_t \in T_D$, and for some pre-defined threshold x ,⁶ then set C rdfs:subClassOf D if $o(T_s, T_t) \leq o(T_t, T_s)$, and set D rdfs:subClassOf C if $o(T_s, T_t) \geq o(T_t, T_s)$.

For the running example, $o(T_{\text{Event}}, T_{\text{Jazz Festival}}) > o(T_{\text{Jazz Festival}}, T_{\text{Event}})$, and therefore obtain Jazz Festival rdfs:subClassOf Event.

Post-processing. For post-processing, the Alignment API is first invoked for finding alignments between the original input ontologies. Those alignments returned with a confidence value of at least 0.95 are kept, and added to the results previously obtained.⁷ Then a reasoner is invoked (in fact, Jena) which finds inferred alignments. E.g., if A is a subclass of B in one of the input ontologies, and an alignment B rdfs:subClassOf

⁶This threshold was typically 0.8 or 0.6 in the experiments in Section Evaluation, where it is discussed how to set suitable thresholds.

⁷0.95 seems to be the lowest threshold generally giving indisputable results.

C has already been found, then the alignment A `rdfs:subClassOf` C is also added, and finding these alignments is done using a reasoner. Finally the alignment results are serialized in the Alignment API format.

The BLOOMS approach as just described makes heavy use of Wikipedia/DBPedia for bootstrapping. It is natural to ask, if Wikipedia could be replaced with something else. In fact, any upper level ontology or thesaurus could be used, and perhaps there are even more options that were not considered. BLOOMS currently uses Wikipedia because it seemed an intuitive choice due to a number of reasons.

- Wikipedia provides wide thematic coverage.
- The Wikipedia category hierarchy is community-built and thus seemed a natural choice for an alignment system for community-built LOD datasets.
- Wikipedia provides a search feature which can be exploited. This search feature makes it possible to naturally include trees in BLOOMS forests which would be difficult to associate with the input concept name in a more controlled setting, e.g., when using an upper level ontology.

For this work the other alternatives have not been systematically investigated. The evaluation in Section 6.4 shows that the current approach using Wikipedia is already rather strong. It is left for future work to investigate to what extent alternatives would bring an increase in performance. It can be hypothesized that alternatives should indeed be very helpful for alignment in more specialized thematic domains, e.g., for life science data in the LOD Cloud. Potential alternatives include the following: Ontologies such as Cyc or SUMO, as used, e.g., in [80]; Thesauri such as WordNet;⁸ Taxonomies created from Wikipedia, such as the one reported in [94]; or efforts like the Open Directory Project⁹ or YAGO [108].

4.3 Evaluation

BLOOMS¹⁰ system has been implemented in Java on top of the Alignment API framework [32]. BLOOMS utilizes the Jena Framework¹¹ for parsing the ontologies, extracting the concepts and for the mentioned

⁸WordNet is used by the Alignment API [36], and thus is indirectly utilized by BLOOMS approach.

⁹<http://www.dmoz.org/>

¹⁰BLOOMS is available from <http://wiki.knoesis.org/index.php/BLOOMS>

¹¹<http://www.openjena.org/>

reasoning step. The input for BLOOMS is two different ontologies serialized using RDF/XML or OWL.

A comprehensive evaluation of BLOOMS has been performed using third party datasets and other state-of-the-art systems in ontology matching. More specifically, BLOOMS has been evaluated BLOOMS in two different ways. Firstly, the ability of BLOOMS to serve as a general purpose ontology matching system was evaluated, by comparing it with other systems on the Ontology Alignment Evaluation Initiative (OAEI) benchmarks.¹² Secondly, BLOOMS was evaluated for the purpose of LOD schema integration and compared it with other systems for ontology matching on LOD schema alignment.

Established in 2004 by leading researchers in the area of ontology matching, the OAEI aims at forging consensus on methods available for schema matching/ontology integration. As a part of this initiative various datasets and reference alignments between these datasets have been made available for evaluating the performance of the participating systems. The systems are evaluated on various parameters such as precision, recall, endurance to lack of structure in the ontologies and absence of properly named concepts.

The initiative consists of various tracks such as a benchmark track, instance matching and oriented matching. The datasets mainly belong to the very narrow domain of bibliographic information with a number of alternative ontologies of the same domain for which alignments are provided. BLOOMS was evaluated on both the benchmark track and the oriented matching track. In the former the task is to identify (only) equivalence relationships. In the latter the task is to identify subclass relationships. The objective of the BLOOMS system is naturally aligned with these two tracks. Furthermore, the OAEI provides with baselines, and results from the previous version of the oriented matching track are available on the web.¹³

In the 2009 initiative, there were five major systems in the oriented matching track: ASMOV [68], CSR [107], RiMOM [75], AROMA [33] and TaxoMAP [52]. RiMOM and AROMA were picked, for the following reasons: (1) RiMOM was the top system in the oriented track in terms of f-measure and available for download. It was one of the consistent performers in the past two years. (2) AROMA ranked second in the 2008 event. (3) Another important factor was the availability of systems for download in order to run experiments on LOD datasets using them.¹⁴ (4) RiMOM and AROMA utilize different techniques and

¹²<http://oaei.ontologymatching.org/>

¹³<http://oaei.ontologymatching.org/2009/results/oriented/>

¹⁴In the OAEI 2009 initiative there were other systems which performed better than RiMOM, namely ASMOV, Lily and CSR. However, ASMOV is a commercial system and the free version runs only on OAEI 2009 datasets and

hence this gives good variety in the techniques utilized for the purpose of matching. RiMOM, in fact, automatically determines which ontology alignment methods to use for a particular matching task, and what kinds of information to use in the similarity calculation and how to combine multiple methods as necessary. AROMA is an ontology matcher which utilizes association rule mining.

In order to achieve more breadth in the evaluation, recent systems which have not participated in the OAEI were also included. OMViaUO [80] utilizes upper level ontologies such as SUMO and DOLCE as semantic bridges in the ontology matching process. S-Match [47] is another novel approach in which semantic correspondences are discovered by computing and returning, (as a result) the semantic information implicitly or explicitly codified in the labels of nodes and arcs.

Some of the systems had tunable parameters. As mentioned in Section 6.3, BLOOMS was used with a threshold value of 0.8 for the ontologies belonging to the same domain, and used a value of 0.6 where one of the ontologies was an abstract ontology such as DBpedia or SUMO. This was done for the following reasons: (1) BLOOMS trees for concepts belonging to the same domain were expected to have higher overlap. (2) Relations between an abstract and a domain specific ontology can be found using a lower overlap. This is because BLOOMS trees constructed for concepts in the domain specific ontology will usually require more nodes to become generic enough in order to match a concept of the more generic ontology.

For RiMOM, while evaluating on LOD datasets, based on the understanding a number of thresholds were specified in the "MatchThreshold" parameter, which range from 0.3 to 0.8. However, the execution with the different parameters always resulted in the same output. On inspection of the results, it was found that there were entries with threshold values as low as 0.01 in the output file.

For AROMA, a threshold of 0.6 for "lexicalThreshold" was utilized. While parameters below 0.5 were too low and resulted in very poor precision, higher thresholds such as 0.8 resulted in identification of very few results. If guidelines were available for deciding the thresholds, it might have been able to tune the system in a better way.

S-Match GUI does not provide functionality for tuning threshold.

therefore cannot be used on LOD datasets. CSR is not available for download and requests for an evaluation copy remained unanswered. TaxoMAP and Lily did not work due to platform incompatibility issues, and support requests were not answered in time.

Table 4.1: Results on the oriented matching track. Results for RiMOM and AROMA have been taken from the OAEI 2009 website. Legends: Prec=Precision, Rec=Recall, A-API=Alignment API, OMV=OMViaUO, NaN=division by zero, likely due to empty alignment.

Ontology Alignment Initiative—Oriented Matching Track												
	A-API		OMV		S-Match		AROMA		RiMoM		BLOOMS	
Test	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec
1XX	0	0	0.02	0.06	0.01	0.71	NaN	0	1	1	1	1
2XX	0	0	0.01	0.03	0.05	0.30	0.84	0.08	0.67	0.85	0.52	0.51
3XX	0.01	0.03	0.02	0.047	0.01	0.14	0.72	0.11	0.59	0.81	1	0.84
Avg.	0.00	0.01	0.02	0.04	0.03	0.38	0.63	0.07	0.75	0.88	0.84	0.78

OMViaUO literature was consulted to get information related to setting suitable thresholds. However, there was no discussion related to this. Further, with respect to the Alignment API and OMViaUO, altering the threshold values (even to 0) did not result in any significant improvement of results on LOD datasets. For the Alignment API and OMViaUO the threshold was kept at 0.5 to achieve an optimum balance between precision and recall.

4.3.1 Evaluation: Ontology Alignment Evaluation Initiative Oriented Track.

In order to test the quality of mappings generated using BLOOMS, the system was run on the oriented datasets using the reference alignment and compared its performance with the other systems mentioned above. Table 4.1 presents the results on the oriented matching track of the OAEI. The different tests 1XX, 2XX, and 3XX comprise of matching a single source ontology (101) to other ontologies beginning with the prefix digit of the test. Thus, test 1XX comprises of matching ontology 101 to ontologies 101, 103, and so forth. Similarly 2XX comprises of matching ontology 101 to ontologies 201, 202, and so forth. Unlike the ontologies used in the tests 1XX and 2XX which are created by the organizers, the test 3XX comprises of ontologies which have been created by other organizations and are used in the real world. The precision and recall figures were computed using the baselines and results made available on the OAEI website.

In the oriented matching track, BLOOMS along with RiMOM provided superior results in the test 1XX. For the test 2XX, all systems including BLOOMS show a drop in the performance. The reasons for this drop might be the following. (1) Some ontologies in test 2XX contain concepts from French. Thus

systems which rely on lexico-syntactic tools obviously have difficulties with these ontologies.¹⁵ (2) Some of these ontologies consist of concepts with random names where the matching has to be done on the basis of structure alone.

For the test 3XX, BLOOMS outperforms the other systems in its recall without comprising on its precision. The reasons for the superior performance of BLOOMS could be the following: (1) Wikipedia has a large number of articles with a rich category hierarchy in which the articles and categories summarize the concepts mentioned in the real world ontologies. (2) The ontologies in these tests are of related domains (e.g. Scientific Publishing) and therefore, require a higher overlap between the BLOOMS trees for two concepts to be related. A higher overlap threshold enforces that the concepts and their corresponding BLOOMS trees have to be very similar. This reduces the number of false positives. (3) The mentioned invocation of a reasoner allows to identify some of the concepts which otherwise have to be found using the structure of the ontology.

The other systems (besides RiMOM) suffer from poor precision and recall due to a variety of reasons. (1) A number of systems such as OMViaUO generate only equivalence mappings. In the oriented matching track, the provided reference alignments consist mainly of subsumption relationships. (2) While S-Match provides good results for the recall, its precision is affected by a plethora of results which are generated for the ontologies. S-Match produces two different output files. The "default results" files were utilized, since it gives a larger number of results. The other file "minimal results" produces a very small set of results, which one could expect to have a higher precision but lower recall, but this is not necessarily the case. For example, for matching ontologies 101 and 103, S-Match produced 267 results in the default file (precision: 0.46; recall: 0.50), and 57 in the minimal file (precision: 0; recall: 0). (3) OMViaUO could not produce satisfactory results due to poor matching performance. The reason for this could be the absence of required ontological concepts in WordNet and in the upper level ontologies utilized by OMViaUO. (4) The Alignment API also suffered from poor precision and recall due to reasons similar to those for OMViaUO. (5) It is assumed AROMA suffers from poor results due to difficulties in identifying association rules related to the ontologies.

¹⁵In future investigations, one could attempt to exploit the fact that Wikipedia is available in many languages, and that the different-language versions are in fact interlinked.

Table 4.2: Comparison of various systems on the benchmark track. Results for RiMOM and AROMA have been reused from the OAEI 2009 website. Legends: Prec=Precision, Rec=Recall

Ontology Alignment Initiative—Benchmark Track												
	S-Match		OMViaUO		Alignment API		BLOOMS		AROMA		RiMoM	
Test	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Recall	Prec	Rec
1XX	0.11	1	0.26	0.37	0.59	0.96	0.71	1	1	1	1	1
2XX	0.1	0.2	0.21	0.31	0.3	0.54	0.38	0.49	0.88	0.65	0.93	0.81
3XX	0.1	0.2	0.28	0.28	0.45	0.77	0.62	0.84	0.80	0.76	0.81	0.82
Avg.	0.1	0.46	0.25	0.33	0.45	0.76	0.57	0.78	0.88	0.81	0.91	0.88

4.3.2 Evaluation: Ontology Alignment Evaluation Initiative Benchmark Track

To test the quality of mappings generated using BLOOMS, it was ran it on the benchmark datasets using the reference alignment and compared its performance with the other systems mentioned above. Table 4.2 presents the results on the benchmark track of the ontology alignment initiative. As in the oriented matching track, the different tests 1XX, 2XX and 3XX comprise of matching a source ontology to other ontologies beginning with the prefix digit of the test. This test utilizes a larger number of ontologies than the oriented matching track. However, to a large extent the ontologies involved are identical.

In the benchmark track, BLOOMS is able to retrieve all results in 1XX, however, it results in loss of precision. In the 1XX track, the other systems gave varying performances. RiMOM and AROMA are impressive with their excellent precision and recall, whereas S-Match and OMViaUO suffer from retrieval of few and incorrect results.

BLOOMS does a better job in 3XX than 2XX due to the involvement of real world ontologies. It ranks right behind RiMOM and AROMA in its recall and does a decent job with respect to precision. The Alignment API does a significantly better job in retrieving the results and matching the ontologies, probably due to the fact, that this track involves finding equivalence relations between ontological concepts. The reasons for poor performance of the other systems are identical to those in the oriented track.

For the 3XX test, BLOOMS outperforms RiMOM and the other systems in finding the correct results. However, the increase in recall goes with a dip in precision. AROMA performs the best in terms of precision.

4.4 Related Work

In this section, the related work in ontology matching and LOD integration is being reported. To the best of our knowledge, this is the first work which exploits a generic and noisy categorization system such as Wikipedia in the context of ontology matching. In the context of ontology matching, traditional techniques rely on three different conceptual approaches: (1) Use of linguistic techniques such as string analysis. (2) Use of structural information such as sub-class relationships. (3) Use of thesauri or upper level ontologies such as WordNet, DOLCE, SUMO and Cyc. There are various tools and techniques which have been developed using a combination of these three ideas. In [23, 38] the authors present one of the best available compilations of the tools and techniques in the area of ontology matching. At a higher level, BLOOMS system utilizes a combination of these three techniques for the purpose of ontology matching. The ontology matching portal¹⁶ gives a good review of the state of the art research in this area, some of which have been mentioned in the evaluation section. In the past, Wikipedia categorization has been utilized for other purposes such as for creating [94] and restructuring taxonomies [93]. Previously, Wikipedia has been utilized for mapping Cyc onto Wikipedia articles describing corresponding concepts [82].

Although ontology matching in Semantic Web is a relatively new area, the field of schema matching has a long tradition in Computer Science. Schema matching has applicability in diverse areas such as databases, web service composition and XML Schema matching. Previously, authors have presented a taxonomy that covers popular approaches in database schema matching [97]. Further, the authors have investigated algorithms for generic schema matching outside of any particular data model or application [77]. In the work, they also present the *Cupid* algorithm, that discovers mappings between schema elements based on their names, data types, constraints, and schema structure, using an array of techniques.

There have been multiple efforts towards the automated integration of LOD datasets at the instance level. However, there are few notable efforts at linking these datasets at the schema level. Recently, ontology schema matching was attempted to improve instance co-reference resolution [87]. Although the work helps in cleaning up the data and improving the quality of links at the instance level, the issue of identifying appropriate relationships at the schema level has not been addressed. The voiD Framework [1] provides a vocabulary as a common format for expressing instance level relationships between LOD datasets. Comple-

¹⁶<http://www.ontologymatching.org/>

mentary to this, the SILK Framework [116] automates the process of link discovery of LOD datasets at the instance level.

An integration of SUMO with DBpedia and YAGO at the schema level has been done manually.¹⁷ While the end goals of BLOOMS and this effort are identical, expanding the manual approach to other datasets will require a significant efforts. Similarly, DBpedia has been linked to other ontologies such as OpenCyc, UMBEL and YAGO. At the schema level, a notable effort for creating a unified reference point for LOD schemas is UMBEL [11], which is a coherent framework for ontology development which can serve as a reference framework. Therefore, it helps in checking for coherence between ontologies that are linked to the UMBEL framework.

¹⁷<http://bit.ly/d8vzvR>

Contextual Ontology Alignment of LOD

Ontology alignment is an important requirement for fulfilling the vision and idea of semantic web. While a significant amount of progress has been made towards this idea, most of the work including BLOOMS [61] are focused towards matching the entities without taking the context in which they occur into account. For example, an entity 'cricket' can be matched to an *insect* or a *sport* depending on the other entities surrounding cricket.

In this chapter we present our approach for the alignment of entities based on their context, namely BLOOMS+ [67].

5.1 Introduction

The Linked Open Data (LOD) is a major milestone towards realizing the Semantic Web vision. Like mentioned earlier, a key differentiator of LOD from previous approaches is that data providers are actually creating links across these data sets, which has led to a number of innovative applications spanning multiple, disparate information sources [17]. One missing facet of LOD so far is that these ever-growing ontologies are linked to each other mainly at the instance-level. There are very few schema-level linkages – i.e. links between class hierarchies such as `rdfs:subClassOf` relations.

A number of researchers [63, 61, 91]¹ have argued that without schema-level linkages the LOD cloud will not have semantic-enough information to enable more ambitious, reasoning-based applications of Semantic Web such as Question Answering and Agent-based information brokering. Existing efforts to de-

¹<http://semtech2010.semanticuniverse.com/sessionPop.cfm?confid=42&proposolid=2854>

velop these types of applications primarily utilize manually created schema-level links between LOD ontologies. For example, *FactForge* enables querying across various LOD ontologies, and utilizes manually developed schema-level mappings of LOD ontologies to an upper level ontology called *Proton* [31].

While definitely useful, the manual creation of schema-level mappings across LOD ontologies is not a viable solution given the size of the LOD and the rate at which it is growing. A more automated solution is needed in order for applications such as *FactForge* to effectively scale to (and keep up with) the size of LOD. To this effect, in the previous chapter a solution, called Bootstrapping-based Linked Open Data Ontology Matching System (BLOOMS) [61] was introduced for automatically finding schema-level links between LOD ontologies. The previous solution performed well on this task compared to existing solutions such as [46, 33, 75, 80], but there is significant room for improvement.

In this chapter, a solution called BLOOMS+ is presented which extends the previous solution in two significant ways. BLOOMS+ 1) uses a more sophisticated metric to determine which classes between two ontologies to align, and 2) BLOOMS+ considers contextual information to further support (or reject) an alignment. The chapter also presents a comprehensive evaluation of BLOOMS+ using schema-level mappings from various LOD ontologies to Proton (an upper level ontology), created manually by human experts. The chapter shows that BLOOM+ performed well on this task. BLOOMS+ is also compared to existing ontology alignment solutions (including our previously published work on BLOOMS) on this same task, and it is shown that BLOOMS+ outperformed these solutions. Finally, an ablation study is presented, which shows why BLOOMS+ performed well.

The rest of the chapter is organized as follows. The knowledge requirements for BLOOMS+ are first presented, and explanation for why Wikipedia was selected to satisfy these requirements. Then the BLOOMS+ approach is presented, followed by a comprehensive evaluation of BLOOMS+ and existing solutions. Finally, the related work is presented along with conclusions and future work.

5.2 Knowledge Requirements

BLOOMS+ requires a knowledge source to align two ontologies. The minimum requirements for this knowledge source are:

1. The knowledge source is organized as a class hierarchy where links between classes in this hierarchy capture super and subclass relationships.
2. The knowledge source covers a wide range of concepts and domains, so it can be widely applicable – especially given the wide range of domains covered by the LOD Cloud.

Many knowledge sources – such as WordNet [39], FrameNet [6], SNOMED [28], etc. – satisfy the first requirement, but they fail to satisfy the second. For example, many classes in WordNet and FrameNet are very generic, and hence may have limited utility when aligning domain specific LOD schemas such as Music and Census. SNOMED, on the other hand, captures classes specific to the medical domain, and can be useful for aligning life science LOD schemas. However, it will have limited utility in aligning LOD schemas outside of life science.

BLOOMS+ uses Wikipedia – in particular the category hierarchy in Wikipedia. Although the Wikipedia category hierarchy is not a formal class hierarchy, it still reflects a taxonomy structure. Wikipedia categories roughly correspond to classes in a class hierarchy, and the super and subcategory relationships between these categories roughly correspond to super and subclass relationships. Wikipedia also covers a wide range of categories (over 10 million categories), across many domains. This satisfied the second requirement. Moreover, previous research [61] has shown that the Wikipedia category hierarchy is effective in aligning LOD schemas.

5.3 Approach

BLOOMS+ aligns two ontologies through the following steps. BLOOMS+ first uses Wikipedia to construct a set of category hierarchy trees for each class in the source and target ontologies. BLOOMS+ then determines which classes to align by extending BLOOMS in two significant ways. BLOOMS+ 1) uses a more sophisticated measure to compute the similarity between source and target classes based on their category hierarchy trees; and 2) computes the contextual similarity between these classes to further support (or reject) an alignment. Finally, BLOOMS+ aligns classes with high similarity based on the class and contextual similarity.

5.3.1 Construct BLOOMS+ Forest

BLOOMS+ constructs a set of category hierarchy trees – we call a *BLOOMS+ Forest* F for each class C from the source and target ontologies. For each C , BLOOMS+ tokenizes (and stems) the name of C , and removes stop words from the name.

BLOOMS+ uses the resulting terms as a search string to retrieve relevant Wikipedia pages using Wikipedia search web service.² BLOOMS+ treats each page as a possible sense s_i of C and constructs a category hierarchy tree we call a BLOOMS+ tree T_i – for s_i via the following steps.

1. The root of the tree is s_i .
2. The immediate children of s_i are all Wikipedia categories that s_i belongs to.
3. Each subsequent level includes all unique, direct super categories of the categories at the current level.

BLOOMS+ imposes a limit on the depth of the tree being constructed, and defaults this limit to 4. Based on empirical observation depths beyond 4 typically include very general categories (e.g. “Humanities”), which are not useful for alignment. The resulting tree is then added to F .

5.3.2 Compute Class Similarity

BLOOMS+ compares each class C in the source ontology with each class D in the target ontology to determine their similarity. This is done by comparing each $T_i \in F_C$ with each $T_j \in F_D$ where F_C and F_D are the BLOOMS+ forests for C and D respectively. For each source tree T_i , BLOOMS+ determines its overlap with the target tree T_j .

However, simply counting the number of common nodes the approach used by BLOOMS is insufficient for the following reasons:

- Common nodes that appear deeper in the tree are more generic (and hence less discriminative). They can appear in many BLOOMS+ trees, which can result in false alignments. These nodes should

²<http://en.wikipedia.org/w/api.php>

be given less importance when computing the overlap between two trees (and hence the similarity between two classes).

- A large tree can be unfairly penalized because it must have more nodes in common with another tree in order to have a high similarity score. Hence, we need to avoid bias against large trees when computing the overlap.

To address these issues, BLOOMS+ uses the following equation to compute the overlap between two BLOOMS+ trees (and hence the similarity of their corresponding classes).

$$Overlap(T_i, T_j) = \frac{\log \sum_{n \in T_i \cap T_j} (1 + e^{d(n)^{-1}-1})}{\log 2|T_i|} \quad (5.1)$$

where $n \in T_i \cap T_j$ are the common nodes between the source and target tree; and $d(n)$ is the depth of a common node n in T_i . The exponentiation of the inverse depth of a common node gives less importance to the node if it is generic, and the log of the tree size avoids bias against large trees. This equation ranges from 0.0 to 1.0 where 0.0 indicates no similarity and 1.0 indicates maximum similarity.

For example, let's assume BLOOMS+ needs to determine whether to align the source class *RecordLabel* from DBpedia with the target class *MusicCompany* from Proton. BLOOMS+ first constructs the BLOOMS+ forests for *RecordLabel* and *MusicCompany*, and Figure 5.3.2 shows a BLOOMS+ tree from each forest. BLOOMS+ then identifies the common nodes between these trees, and the depth of these nodes in the tree for the source class (see Table 5.1).

Finally, the class similarity (see above equation) between *RecordLabel* and *MusicCompany* w.r.t the two BLOOMS+ trees in Figure 5.3.2 is 0.79.

5.3.3 Compute Contextual Similarity

BLOOMS+ computes the contextual similarity between a source C and target D class to further determine whether these classes should be aligned. A good source of contextual information is the superclasses of C and D from their respective ontologies. If these superclasses agree with each other, then the alignment between C and D is further supported and hence should be given more preference. Otherwise, the alignment

(a) BLOOMS+ tree for *RecordLabel* with sense *Record Label* (b) BLOOMS+ tree for *MusicCompany* with sense *Music Industry*Figure 5.1: BLOOMS+ trees for *Record Label* 5.1(a) and *Music Company* 5.1(b)

should be penalized. For example, the class *Jaguar* might be aligned to the class *Cat*, which seems like a reasonable alignment. However, if *Jaguar* has superclasses such as *Car* and *Vehicle*, and *Cat* has superclasses such as *Feline* and *Mammal*, then the alignment should be penalized because its contextual similarity is low.

BLOOMS+ implements the intuition above in the following way. For each pair wise class comparison (C, D) , BLOOMS+ retrieves all superclasses of C and D up to a specified level, which BLOOMS+ defaults

Table 5.1: Common nodes between the two trees in Figure 5.3.2, and their depth. The first column gives the common nodes between the two trees rooted at *Record Label* and *Music Industry*. The second column gives the depth (the distance from root) of these nodes in the BLOOMS+ tree rooted at *Record Label* – i.e. the source tree.

Common Nodes	Node Depth
Music_industry	1
Music; Industries; Cultural_economics	2
Industry; Other_special_topics_(economics); Cultural_studies; Economic_systems; Entertainment; Performing_arts; Sound	3

to 2. The two sets of superclasses – we’ll refer to as $N(C)$ and $N(D)$ – are the neighborhoods of C and D respectively.

For each BLOOMS+ tree pair (T_i, T_j) between C and D , BLOOMS+ determines the number of superclasses in $N(C)$ and $N(D)$ that are supported by T_i and T_j respectively. A superclass $c \in N(C)$ is supported by T_i if either of the following conditions are satisfied:

- The name of c matches a node in T_i .³
- The Wikipedia article (or article category) corresponding to c – based on a Wikipedia search web service call using the name of c – matches a node in T_i .

The same applies for a superclass $d \in N(D)$.

BLOOMS+ computes the overall contextual similarity between C and D with respect to T_i and T_j using the harmonic mean, which is instantiated as:

$$CSim(T_i, T_j) = \frac{2R_C R_D}{R_C + R_D} \quad (5.2)$$

where R_C (and R_D) are the fraction of superclasses in $N(C)$ (and $N(D)$) supported by T_i (and T_j). The harmonic mean was chosen to emphasize superclass neighborhoods that are not well supported (and hence should significantly lower the overall contextual similarity).

³This match is defined as either a direct string match or a substring match.

Returning to the example, BLOOMS+ needs to compute the contextual similarity for *RecordLabel* and *MusicCompany*. Assuming a level of 2, the neighborhood of *RecordLabel* includes the DBpedia superclasses of *Company* and *Organization*. Both superclasses are supported by the BLOOMS+ tree for *RecordLabel* (see Figure 5.1(a)), so $R_{RecordLabel}$ is $\frac{2}{2}$. Similarly, the neighborhood of *MusicCompany* includes the Proton superclasses of *CommercialOrganization* and *Organization*. Both superclasses are supported by the BLOOMS+ tree for *MusicCompany* (see Figure 5.1(b)), so $R_{MusicCompany}$ is also $\frac{2}{2}$. Finally, the overall contextual similarity (see above equation) is 1.0, so BLOOMS+ should give more preference to this alignment.

5.3.4 Compute Overall Similarity

BLOOMS+ computes the overall similarity between classes C and D w.r.t. BLOOMS+ trees T_i and T_j by taking the weighted average of the class (see Section 5.3.2) and contextual (see Section 5.3.3) similarity.

$$O(T_i, T_j) = \frac{\alpha \text{Overlap}(T_i, T_j) + \beta C\text{Sim}(T_i, T_j)}{2} \quad (5.3)$$

where α and β are weights for the concept and contextual similarity respectively. BLOOMS+ defaults both α and β to 1.0 to give equal importance to each component.

BLOOMS+ then selects the tree pair $(T_i, T_j) \in F_C \times F_D$ with the highest overall similarity score and if this score is greater than the alignment threshold H_A , then BLOOMS+ will establish a link between C and D . The type of link is determined as follows:

- If $O(T_i, T_j) = O(T_j, T_i)$, then BLOOMS+ sets C owl:equivalentClass D .
- If $O(T_i, T_j) < O(T_j, T_i)$, then BLOOMS+ sets C rdfs:subClassOf D .
- Otherwise, BLOOMS+ sets D rdfs:subClassOf C .

Returning to our running example, the overall similarity score between *RecordLabel* and *MusicCompany* is 0.895 (i.e. $\frac{0.79+1.0}{2}$), and BLOOMS+ will establish a link between these classes – assuming the alignment threshold is 0.5. Finally, BLOOMS+ sets *RecordLabel* rdfs:subClassOf *MusicCompany* because $O(T_{\text{Music Industry}}, T_{\text{Record Label}}) > O(T_{\text{Record Label}}, T_{\text{Music Industry}})$.

5.4 Evaluation

The following claims were evaluated to show that the approach (i.e. BLOOMS+) is effective for ontology alignment over LOD schemas.

Claim 1: BLOOMS+ can outperform state-of-the-art solutions on the task of aligning LOD ontologies.

Claim 2: BLOOMS+ performs well because it accounts for two critical factors when computing the similarity between two classes – 1) the importance of common nodes between the BLOOMS+ trees of the two classes, and 2) bias against large trees.

Claim 3: The performance of BLOOMS+ can be further improved by using contextual information.

5.4.1 Data Set

A real world data set was used for the evaluation. This data set contains schema-level mappings from three LOD ontologies to Proton, an upper level ontology, with over 300 classes and 100 properties, designed to support applications such as semantic annotation, indexing, and search[109]. The three LOD ontologies include:

- **DBpedia:**⁴ The RDF version of Wikipedia, created manually from Wikipedia article infoboxes. DBpedia consists of 259 classes ranging from general classes (e.g. Event) to domain specific ones (e.g. Protein).
- **Freebase:**⁵ A large collection of structured data collected from multiple sources such as Wikipedia, Chefmoz, and MusicBrainz. Freebase consists of over 5 million topics and entities, classified into a class hierarchy.
- **Geonames:**⁶ A geographic data set with over 6 million locations of interest, which are classified into 11 different classes.

⁴http://downloads.dbpedia.org/3.5.1/dbpedia_3.5.1.owl.bz2

⁵<http://www.freebase.com/schema>

⁶<http://geonames.org>

Table 5.2: Sample mappings of LOD ontologies to PROTON.

Ontology	Class	PROTON Class	Relationship
DBpedia	OlympicResult	Situation	subClassOf
Geonames	Class	LandRegion	subClassOf
Freebase	Event	Event	equivalentClassOf

These mappings were systematically created by Knowledge Engineers (KEs) [31] at OntoText for a real world application called *FactForge*⁷, which enables SPARQL query over the LOD cloud. The KEs created these mappings, i.e. equivalence and subclass relationships between LOD and Proton classes, based on the definition of the classes and their usage. A total of 544 mappings were created from the three LOD ontologies to Proton (373 for DBpedia, 21 for Geonames, and 150 for Freebase). Table 5.2 shows examples of these mappings.

These mappings provide a good gold standard for our evaluation because:

- The mappings were created by an independent source for a real world use case – unlike existing benchmarks which were created primarily for evaluation purposes. Hence, these mappings reflect the types of relationship that are needed in practice.
- The mappings were created by knowledge engineers through a systematic process [31] and hence are of high quality.
- The mappings cover a diverse set of LOD ontologies. For example, DBpedia and Freebase cover diverse domains such as entertainment, sports, and politics. While Geonames covers only geographic information.

5.4.2 Experimental Setup

To evaluate Claim 1, the precision and recall of the mappings was measured from the three LOD ontologies to Proton generated by BLOOMS+. To obtain these measures, BLOOMS+ was applied to each LOD-Proton ontology pair to generate mappings whose overall similarity exceeded an alignment threshold of 0.85 (see Section 5.3.4). This threshold was defined by systematically analyzing which threshold level produced the

⁷<http://factforge.net/>

best f-measure score. The resulting mappings was then compared for each LOD-Proton ontology pair to their respective gold standard, and said that a mapping between two classes is correct if the gold standard also established a mapping between these two classes using the same relationship i.e. equivalence or subclass. Finally, precision is defined as the number of correct mappings over the total number of mappings generated by BLOOMS+, and recall as the number of correct mappings over all mappings in the gold standard.

The performance of BLOOMS+ was also compared to existing solutions that performed well for LOD ontology alignment, as reported in [61]. These solutions include:

- **BLOOMS:** This is the solution that BLOOMS+ extends [61].
- **S-Match:** This solution utilizes three matching algorithms – basic, minimal, and structure preserving – to establish mappings between the classes of two ontologies [46].
- **AROMA:** This solution utilizes the association rule mining paradigm to discover equivalence and subclass relationships between the classes of two ontologies [33].

To ensure a fair comparison, the above methodology was used to measure precision and recall for each solution, and to define the alignment threshold. The best alignment threshold for BLOOMS is 0.6. The performance of AROMA was not affected by the alignment threshold. It had identical performance for all threshold levels between 0.1 to 1.0. S-Match does not support an alignment threshold. Instead, it returns two sets of mappings – 1) a minimal set and 2) a complete set, which can be derived from the minimal one. Both sets of results have been reported in this evaluation.

To evaluate Claims 2 and 3, a version of BLOOMS+ was created without contextual information known as *BLOOMS+ NO-CONTEXT*. The only difference between BLOOMS+ NO-CONTEXT and BLOOMS is the measure used to compute the similarity between two classes (and hence allows to evaluate Claim 2). The only difference between BLOOMS+ NO-CONTEXT and BLOOMS+ is the use of contextual information (and hence allows to evaluate Claim 3). The above methodology was used to measure precision and recall for BLOOMS+ NO-CONTEXT, and the alignment threshold was set to 0.85. The evaluation components related to this work are available for download on BLOOMS+ project page.⁸

⁸<http://wiki.knoesis.org/index.php/CBLOOMS>

Table 5.3: Results for various solutions on the task of aligning LOD schemas to PROTON. Legend: S-Match-M=Result of S-Match Minimal Set, S-Match-C=Result of S-Match Complete Set, Prec=Precision, Rec=Recall, F=F-Measure PRO=PROTON Ontology, FB=Freebase Ontology, DB=DBpedia Ontology, GEO=Geonames Ontology

Linked Open Data and Proton Schema Ontology Alignment											
	DB-PRO			GEO-PRO			FB-PRO			Overall	
System	Rec	Prec	F	Rec	Prec	F	Rec	Prec	F	Rec	Prec
AROMA	0.19	0.59	0.28	0.04	$\frac{8}{1000}$	0.01	0.31	0.49	0.38	0.22	0.37
S-Match-M	0.26	0.05	0.08	0.04	$\frac{6}{1000}$	0.01	0.2	0.05	0.08	0.23	0.05
S-Match-C	0.33	$\frac{3}{1000}$	$\frac{6}{1000}$	0.04	0.009	0.01	0.3	0.4	0.34	0.31	$\frac{4}{1000}$
BLOOMS	0.48	0.19	0.27	0.04	$\frac{6}{1000}$	0.01	0.28	0.32	0.3	0.42	0.19
BLOOMS+	0.77	0.59	0.67	0.04	$\frac{5}{1000}$	0.01	0.48	0.65	0.55	0.66	0.45
No Context											
BLOOMS+	0.73	0.90	0.81	0.04	$\frac{5}{1000}$	0.01	0.49	0.59	0.54	0.63	0.55

Table 5.4: Sample of **correct** mappings from LOD ontologies to PROTON generated by BLOOMS+.

Ontology	LOD Class	PROTON Class	Relationship
DBpedia	RecordLabel	MusicCompany	subClassOf
Geonames	Country	Nation	equivalentClassOf
Freebase	Military_command	Position	subClassOf

5.4.3 Results and Discussion

Table 5.3 shows the results for all solutions evaluated. Table 5.4 and Table 5.5 show examples of correct and incorrect mappings respectively generated by BLOOMS+ from the three LOD ontologies to Proton.

BLOOMS+ performed significantly better than all other solutions in the evaluation on both precision and recall for two LOD-Proton ontology pairs ($p < 0.01$ for χ^2 test in all cases). BLOOMS+ performed well because it utilizes 1) a rich knowledge source – i.e. Wikipedia – to determine the similarity between the classes of two ontologies and 2) contextual information from both Wikipedia and the ontologies being aligned. Hence, these results support the first claim that BLOOMS+ can outperform the state-of-the-art on the task of aligning LOD ontologies.

Interestingly, no solution performed well on aligning Geonames with Proton. The only mapping found by BLOOMS+ (and the other solutions) is the class *Country* in Geonames is equivalent to the class *Nation* in Proton. The key reasons for the poor performance include: 1) Geonames has a small number of classes (and

Table 5.5: Sample of **incorrect** mappings from LOD ontologies to PROTON generated by BLOOMS+.

Ontology	LOD Class	PROTON Class	Relationship
DBpedia	Writer	Message	subClassOf
Geonames	Feature	Art	subClassOf
Freebase	Military_command	Event	subClassOf

hence very limited contextual information) and 2) the names of the classes in Geonames are often vague and ambiguous (e.g. *Code* and *Feature*), which made it difficult to compute their similarity.

BLOOMS+-NO-CONTEXT performed significantly better than BLOOMS w.r.t the overall precision and recall ($p < 0.01$ for χ^2 test on both precision and recall). This improvement can be attributed to the only difference between the two solutions. BLOOMS+-NO-CONTEXT uses a more sophisticated measure to compute the similarity between two classes. This measure considers the importance of common nodes between the BLOOMS+ trees of two classes, and avoids bias against large trees. This result supports the second claim that BLOOMS+ performs well because it considers the importance of common nodes and avoids bias against large trees when computing the similarity between two classes.

BLOOMS+ performed significantly better than BLOOMS+-NO-CONTEXT w.r.t to the overall precision ($p < 0.01$ for χ^2 test). Although BLOOMS+ had lower overall recall, this difference was not statistically significant according to the χ^2 test. Moreover, BLOOMS+ had a higher overall f-measure score. This result can be attributed to the only difference between these two solutions. BLOOMS+ uses contextual information, and BLOOMS+-NO-CONTEXT does not. Hence, this result supports the third claim that the use of contextual information can further improve performance – in particular precision and f-measure.

5.5 Related Work

To the best of our knowledge, the only other work which exploits contextual information for the purpose of ontology matching has been described in [37]. However, their approach is different from BLOOMS+ as they rely on background knowledge from online ontologies, whereas we rely on a noisy loose categorization of Wikipedia for performing the contextual match. Further, their process relies on identification of contextual relationship using the relationships encoded in the ontologies.

Research in the area of 'Ontology Matching' is very closely related to this body of work. In [38, 23] the authors present a survey in the area of ontology matching.⁹ The survey work also categorizes the techniques on the basis of external knowledge source utilized by ontology matching systems. While typically, systems utilize a structured source of information such as dictionaries or upper level ontologies, In previous work in [61] an approach has been presented which exploits a generic and noisy categorization system such as Wikipedia in the context of ontology matching. Previously, Wikipedia categorization has been utilized for creating and restructuring taxonomies [94, 93].

Another body of related work is identification and creation of links between LOD cloud data sets. In [87] ontology schema matching was used to improve instance co-reference resolution. This helps in cleaning up the data and improving the quality of links at the instance level, but the issue of identifying appropriate relationships at the schema level has not been addressed. The void Framework [1] along with the SILK Framework [116] automate the process of link discovery between LOD datasets at the instance level. At the schema level, a notable effort for creating a unified reference point for LOD schemas is UMBEL [11], which is a coherent framework for ontology development which can serve as a reference framework.

5.6 Conclusion

The chapter presented a solution – called BLOOMS+ – for performing ontology alignment. BLOOMS+ has been evaluated using schema-level mappings from three LOD ontologies to Proton – created manually by human experts for a real world application called *FactForge* – and showed that BLOOMS+ performed well on this task. State-of-the-art ontology alignment solutions were also applied (including BLOOMS) to this same task, and showed that BLOOMS+ significantly outperformed these solutions on both precision and recall. The chapter also showed that the solution performed well because:

- BLOOMS+ uses a rich knowledge source – i.e. Wikipedia – to determine the similarity between the classes of two ontologies;

⁹The ontology matching portal at <http://www.ontologymatching.org/> gives a good review of the state-of-the-art research in this area

-
- BLOOMS+ accounts for two critical factors when computing the similarity between two classes – 1) the importance of common nodes between the BLOOMS+ trees of the two classes, and 2) bias against large trees.
 - BLOOMS+ uses contextual information from both Wikipedia and the ontologies being aligned to further support (or reject) an alignment.

Partonomical Relationship Identification on Linked Open Data

The LOD Cloud consists of datasets linked primarily by the owl:sameAs property created by different organizations. This has proven to be useful for a number of use cases [17, 55], which combine data from multiple ontologies. The current mechanism for linking entities across datasets is using the *sameAs* relationship to assert that two entities are the same. The *sameAs* relationship is not sufficient to capture the rich set of relationships between entities. There are a number of other relationships such as partonomy (part-of), and causality [105], whose presence could allow creating even more intelligent applications such as more sophisticated question answering systems like Watson [41]. One of the main reasons why these relationships are not captured is the issue of scale. As there are millions of entities involved, it is a non-trivial task to manually assert these relationships. While there is some level of automation available for creating the *sameAs* links, there is no automation for creating other kinds of relationships [63].

In this chapter, *PLATO* (**P**art-Of relation finder on **L**inked Open **D**Ata **T**Ool)¹ is presented for automatically creating part-of relationship between entities in the LOD cloud.

6.1 Introduction

The part-of relationship has been presented for two reasons: 1) it is a well studied field. In particular the partonomy classification created by Winston [118] is used to guide our work and 2) part-of relationships are

¹<http://wiki.knoesis.org/index.php/PLATO>

freely available on the Web in sources such as Wikipedia. The fundamental premise behind the approach is that the web can be mined to automatically detect part-of relationships between entities. PLATO approach consists of a combination of heuristics for detecting candidate relationships between any two entities. These heuristics range from detecting bi-directionality of links between articles about these entities to ensuring that the involved entities satisfy domain and range constraints of the relevant partonomic relation. The Web is then mined for evidence to support the candidate relationships with the help of pattern based querying. Using this approach, PLATO is able to discover partonomic relationships between entities in the LOD cloud. For example, PLATO was correctly able to discover that Kurt Cobain was a member of the band Nirvana and that Baked Alaska has ice cream as an ingredient. These relationships can prove to be extremely useful for the LOD cloud. For example, consider the following query from the National Geographic Bee, "In which county can you find the village of Crook that is full of lakes?". The answer for this query can be successfully retrieved using information present in the LOD cloud dataset (e.g. Geonames), if part-of relationships have been identified and asserted within and between datasets [66].

The key contributions of this work are: 1) To the best of our knowledge, PLATO is the first effort on the automatic detection of part-of relationships in the context of the LOD cloud. 2) PLATO's approach of mining the Web to detect and validate the relationships for LOD cloud is rather unique and thus extends the existing arsenal of ontology engineering methods. 3) A formal representation of the partonomy classification created by Winston is provided. A comprehensive evaluation is presented in which it automatically detects part-of relationships between hundreds of entities from prominent ontologies in the LOD cloud such as DBpedia and Freebase. The precision and recall for partonomy extraction approach is also presented, and the results show this is a practically useful approach.

The rest of the chapter is organized as follows: In Section 2 Winston's approach to part-of relation and its conversion into an OWL 2 ontology has been presented. In Section 3, the PLATO approach is presented, followed by a comprehensive evaluation. The related work is presented following future work and conclusion.

6.2 Winston's Approach to Part-of

Relationships—Ontologized

All entities are fundamentally part of some other entity. Researchers in a number of areas, including philosophy [118, 5], linguistics [45] and geographical information systems (GIS) [111, 66, 21] have investigated partonomy. The work of identification of partonomic relationships between entities uses well-accepted partonomic relationships, which identify the relationships based on the 'type' of entities involved. The part-whole relation, or partonomy, is an important fundamental relationship which manifests itself across all physical entities such as human made objects (Cup-Handle), social groups (Jurors-Jury) and conceptual entities such as time intervals (5th hour of the day). Its frequent occurrence results in a manifestation of a part-for-whole mismatch and whole-for-part mismatch within many domains, and especially in spatial datasets.

Winston [118] created a categorization of part-whole relations which identifies and covers part-whole relations from a number of domains such as artifacts, geographical entities, food and liquids. It is recognized as one of the most comprehensive categorizations of partonomic relationships, and other work in similar spirit such as [44] analyze his categorization.

Winston's categorization has been created using three relational elements:

1. Functional/Non-Functional (F/NF): Parts are in a specific spatial/temporal relationship with respect to each other and to the whole to which they belong. Example: Belgium is a part of NATO partly because of its specific spatial position.
2. Homeomeric/Non-Homeomeric (H/NH): Parts are the same as each other and as the whole. Example: A slice of a pie is the same as other slices and as the pie itself.
3. Separable/Inseparable (S/IN): Parts are separable/ inseparable from the whole. Example: A card can be separated from the deck to which it belongs.

Table 6.1 illustrates six different types of partonomic relationships based on this categorization, taken from [118], their description using the relational elements and examples of partonomic relationships covered by them.

Category	Description	Example	Text Patterns
Component-Integral Object	Parts are functional, non-homeomeric and separable from the whole.	Handle-Cup	part of, component of
Member-Collection	Parts are non functional, non homeomeric and separable from the whole.	Tree-Forest	member of, part-of
Portion-Mass	Parts are non-functional, homeomeric and separable from the whole.	Slice-Pie	of, part-of
Stuff-Object	Parts are non-functional, non-homeomeric and inseparable from the whole.	Gin-Martini	is partly, made of
Feature-Activity	Parts are functional, non-homeomeric and inseparable from the whole.	Paying-Shopping	has, have
Place-Area	Parts are non-functional, homeomeric and inseparable from the whole.	Everglades-Florida	located in, part-of

Table 6.1: Six type of partonomic relation with relational elements

Using this classification and relational elements, relations between two entities can be marked as partonomic or non-partonomic in nature. If they are partonomic, the category to which they belong can be identified.

In order to use Winston’s approach in a Semantic Web context, which is essentially linguistic in nature, it must be formalize by carrying it over to a Semantic Web ontology language. This categorization is cast into an OWL 2 ontology [58] which can then be used in conjunction with a knowledge base of partonomic (and other) information. In [98] a set of best practices have been laid down to deal with straightforward cases for defining classes involving part-whole relations. However their modeling approach is considerably

less fine-grained than the one in [118] which is followed here.

For this purpose, the following OWL property names are introduced, which correspond to those listed in Table 6.1.

- component-integral object: **po-component**
- member-collection: **po-member**
- portion-mass: **po-portion**
- stuff-object: **po-stuff**
- feature-activity: **po-feature**
- place-area: **po-place**

The **spatially-located-in** is used as the spatial (topological) located-in relationship mentioned in [118], and **part-of** as the generic part-of (part-whole) relation.

The following axioms can then be drawn from [118]. Let $\mathbf{PO} = \{\mathbf{po-component}, \mathbf{po-member}, \mathbf{po-portion}, \mathbf{po-stuff}, \mathbf{po-feature}, \mathbf{po-place}\}$.

(P1) [118, Section 5] For all $R \in \mathbf{PO}$, R is transitive, asymmetric, and irreflexive (i.e., a strict partial order).

(P2) For all $R \in \mathbf{PO}$, $R \sqsubseteq \mathbf{part-of}$. Note that this does *not* imply that **part-of** is transitive, as prescribed in [118].

(P3) **spatially-located-in** is transitive and reflexive. Note that **spatially-located-in** should not be understood to be a subproperty of **part-of** according to [118].

(P4) [118, Section 6] For all $R \in \mathbf{PO}$, we have

$$R \circ \mathbf{spatially-located-in} \sqsubseteq \mathbf{spatially-located-in} \quad \text{and} \\ \mathbf{spatially-located-in} \circ R \sqsubseteq \mathbf{spatially-located-in}.$$

(P5) [118, page 435] For all $R \in \mathbf{PO} \cup \{\text{spatially-located-in}\}$, and all classes C , the first-order predicate logic axiom is presented

$$(\forall x)(\forall y)(R(x, y) \wedge C(y) \rightarrow (\exists z)(R(x, z) \wedge C(z))).$$

Note that this is a tautology.

(P6) [118, page 435] For all $R \in \mathbf{PO} \cup \{\text{spatially-located-in}\}$, and all classes C , the first-order predicate logic axiom is presented

$$(\forall x)(\forall y)(C(y) \wedge (C(y) \rightarrow R(x, y)) \rightarrow R(x, y)).$$

Please note that this is a tautology.

Summarizing, (P1) to (P4) as the following axioms can be presented— (P5) and (P6) are discussed further below.

- For all $R \in \mathbf{PO}$, R is transitive, antisymmetric, and irreflexive.
- For all $R \in \mathbf{PO}$, $R \sqsubseteq \text{part-of}$.
- $\text{spatially-located-in}$ is transitive and reflexive.
- For all $R \in \mathbf{PO}$,

$$R \circ \text{spatially-located-in} \sqsubseteq \text{spatially-located-in} \quad \text{and} \\ \text{spatially-located-in} \circ R \sqsubseteq \text{spatially-located-in}.$$

This results in a total of $3 \cdot 6 + 2 \cdot 6 + 2 + 6 \cdot 2 = 44$ axioms, all expressible in OWL 2.

However, there is a catch. While all these axioms are expressible in OWL 2 (more precisely, in OWL 2 Full), the collection of these ontologies does not constitute a valid OWL 2 DL ontology. The reason for this is that (P1) violates a global constraint on OWL 2 DL ontologies given in [84, Section 11]: A property

cannot be transitive and irreflexive at the same time.² In other words, it cannot be specified as strict partial orders in OWL 2 DL.³ The most straightforward way to fix this, is to drop one of the requirements on R in (P1), and the most obvious candidate would be to drop the irreflexivity axioms. The resulting set of 38 axioms then constitutes a valid OWL 2 DL ontology.

(P5) and (P6) are tautologies in first-order predicate logic, which means that they do not contribute any additional knowledge. As such, they do not need to be added to the ontology.⁴ Note that this does not mean that the observations leading to (P5) and (P6) in [118] are void: We obtain tautologies because the use of OWL suggests a particular type of modeling class membership (called class inclusion in [118]) which is probably not obvious or necessary from a more general, linguistic perspective.

It is possible to partially recover irreflexivity of the $R \in \mathbf{PO}$. One way to do this is to use the DL-safe SWRL rule [59, 72, 85] $R(x, y) \wedge R(y, x) \rightarrow x \neq y$, which expresses the same as irreflexivity, however its application is restricted to known individuals and is thus weaker than (first-order logic) irreflexivity. Another alternative is to use nominal schemas [72, 73], e.g. by means of the axiom⁵

$$\{x\} \sqcap \exists R. \exists R. \{x\} \sqsubseteq \perp$$

which can actually be understood as a macro that results in n OWL 2 DL axioms, where n is the number of known individuals in the knowledge base.⁶ This means that it can incorporate a weak form of irreflexivity in OWL 2 DL without having to use DL-safe SWRL (and software which supports the latter).

There is yet another catch: All properties occurring in the above constructed part-of ontology are complex (i.e., non-simple), and OWL 2 DL has global restrictions on the use of such properties. If this ontology is used in conjunction with a domain ontology, then these global restrictions may be violated.

²A transitive property is complex, and thus not simple. However only simple properties are allowed to be irreflexive.

³Note that transitivity and irreflexivity of a property R imply that R is also antisymmetric (i.e., a strict partial order): Assume R were transitive and irreflexive, but not antisymmetric. Then, because R is not antisymmetric it must have a, b with $R(a, b)$ and $R(b, a)$ and $a \neq b$. But by transitivity of R , it can be obtained $R(a, a)$ from $R(a, b)$ and $R(b, a)$ which is impossible by irreflexivity.

⁴In other words, adding them would accomplish nothing.

⁵Nominal schemas could also be used to directly express the just mentioned DL-safe rule [73]. However, this would result in a more complicated axiom with two nominal schemas, which is less favorable in terms of scalability.

⁶The OWL 2 DL axioms are obtained by *grounding*: Replace $\{x\}$ by all available nominals $\{a\}$, a being a known individual, each such replacement resulting in one OWL 2 DL axiom.

Likewise, usage of properties in OWL 2 DL is globally restricted by the so-called *regularity* condition,⁷ which may also be violated if the part-of ontology is used together with a domain ontology. In a way similar to the irreflexivity issue discussed above, it is possible to recover from this by expressing some (or all) of the axioms in the part-of ontology in weaker form, using DL-safe rules or nominal schemas. How this is best done depends on the domain ontology, but it is always possible in principle, and indeed relatively straightforward.

6.3 Approach

Given a LOD Cloud dataset, the solution – PLATO – automatically enriches it with partonomy properties through four key steps.⁸

First, PLATO generates candidate pairs of entities from the dataset. Second, PLATO generates “hypothesis” of possible partonomy properties – represented as linguistic patterns – for each entity pair. Next, PLATO tests the resulting patterns (and hence hypotheses) in a corpus driven manner. Finally, PLATO asserts only those partonomy properties with strong supporting evidence.

Figure 6.1 depicts the workflow, which is described in more detail in the subsequent sections.

6.3.1 Candidate Generation

Given a LOD Cloud dataset, PLATO generates all possible pairs between the entities in the dataset. However, the number of entity pairs can be extremely large, which can make the subsequent steps intractable. To address this problem, PLATO filters unpromising entity pairs using a simple heuristic—i.e. entities that are strongly associated are more likely to be related via some property than those that are not. PLATO implements this heuristic by exploiting Wikipedia. The references between Wikipedia pages provide a good proxy for association. Moreover, Wikipedia provides comprehensive coverage across diverse domains. For each entity pair, PLATO retrieves the corresponding Wikipedia page of each entity—using the Mediawiki

⁷See “Restriction on the Property Hierarchy” in [84, Section 11].

⁸PLATO follows these same four steps for enriching multiple LOD Cloud datasets. For ease of exposition, PLATO is described in the context of enriching a single dataset.

Figure 6.1: PLATO System Architecture

API⁹—and if these pages refer to each other, then the pair is said to be strongly associated and kept for subsequent processing. Otherwise, the pair is discarded.

For datasets besides DBpedia, such as Freebase, the sameAs links present between DBpedia entity (e.g. dbpedia: Cellulose) and entity of other datasets (e.g. fbase: Cellulose) are used. Then PLATO checks if the any of the entity refers to the other one. For example, if fbase: Chicken links to dbpedia: Salt. This is just a way to reduce the number of candidate pairs and it is possible to use other techniques to generate these pairs. The use of dataset specific heuristics has been used in other tools such as SILK [117], in order to maximize finding relationships between any two datasets. It is possible to replace this module with another heuristics to generate candidate pairs and use the rest of the system without any modifications.

Please note, in principal it is possible to replace the usage of Mediawiki API with entities directly from DBpedia. However, it may result in the loss of some useful candidate pairs as DBpedia captures limited

⁹<http://en.wikipedia.org/w/api.php>

information from Wikipedia. For example, as of 6th February 2012, the DBpedia page for Cellulose does not refer to Carbon. However, the Wikipedia pages for Carbon and Cellulose do refer to each other, thus making them possible candidate pairs for consideration.

For example, given the DBpedia dataset from the LOD Cloud, some of the entity pairs generated by PLATO will include:

- Cellulose, Cell Wall
- Cellulose, Kraft's Food

PLATO retrieves the Wikipedia pages for Cellulose, Cell Wall, and Kraft's Foods. The Wikipedia pages for Cellulose and Cell Wall refer to each other, so this pair is kept. The Wikipedia page for Cellulose refers to the page for Kraft's Foods, due to usage of Cellulose in cheese manufacturing at Kraft's Foods. However, the page for The Kraft's Foods does not refer back to the page for Cellulose. Hence, this pair is considered to be only weakly associated by PLATO, and thus discarded.

6.3.2 Hypothesis Generation

PLATO generates hypotheses of possible OWL partonomy properties (described in Section 6.2) for each entity pair from the previous step. PLATO now determines the type of each entity in the pair using WordNet [39]—a lexical taxonomy that is well suited for this task. Specifically, PLATO retrieves the lexicographer file of the WordNet synset corresponding to each entity to serve as its type.¹⁰ The name of this file has the form *POS.SUFFIX* where *POS* is the part-of-speech (i.e. noun, verb, adv, or adj) and *SUFFIX* is the broader group that the synset (and hence entity) belongs to (e.g. animal, plant, etc.). For example, given the entity pair (*Cell Wall*, *Cellulose*), lexicographer files of the synsets corresponding to these entities are both *noun.body*.

PLATO uses this information to determine the applicable OWL partonomy properties. These properties are captured from Winston's taxonomy of part-whole relations [118] (see Section 6.2), which was chosen for the following reasons:

¹⁰If a WordNet synset cannot be found for an entity, then PLATO will generalize the entity by looking up its superclass in DBpedia using the JENA ARQ API (<http://openjena.org/>).

- Winston’s taxonomy is well-established and widely accepted.
- Winston provides guidelines on what types are applicable to each part-whole relationship—e.g. Winston’s *Place-Area* relationship applies to only areas, places, and locations. These guidelines can be captured as domain-range axioms for each corresponding OWL partonomy property.
- Winston suggests linguistic cues for each part-whole relationship, which PLATO can use to generate linguistic patterns.

If *POS* is not a noun or verb, then PLATO discards the entity pair because Winston’s relationships apply to only nouns and verbs. If so, then PLATO uses the SUFFIX to determine the OWL partonomy properties that are applicable based on their domain and range. Returning to our example, the OWL properties of *po-component* and *po-stuff*—corresponding to Winston’s *Component-Integral-Object* and *Stuff-Object* relationships respectively—are applicable because the SUFFIXES of Cell Wall and Cellulose satisfy the domain and range of these properties.

Finally, PLATO generates linguistic patterns for each applicable property based on linguistic cues suggested by Winston. For example, the linguistic cues for *po-stuff* include “is made of” and “is partly.” From these cues, the following linguistic patterns are generated for (*Cell Wall*, *Cellulose*):

- Cell Wall is made of Cellulose
- Cellulose is made of Cell Wall
- Cell Wall is partly Cellulose
- Cellulose is partly Cell Wall

These patterns serve as hypotheses to be validated in the next step.

6.3.3 Hypothesis Testing

PLATO tests the lexical patterns for each entity pair in a corpus-driven manner. PLATO uses the Web as the corpus because of its coverage, and uses publicly available search APIs to access its contents. Specifically,

PLATO uses the Bing Search API 2.0¹¹ because it allows unlimited searches.

For each pattern generated for an entity pair, PLATO executes a search of the pattern using the BING API, and takes the top N search results (i.e. URLs for the top N webpages) returned by BING. N can be adjusted by the user; and PLATO sets the default value of N to 50, which we found to produce good results empirically. For each resulting URL, PLATO fetches the page it points to—using off-the-shelf crawling and html parsing technologies, e.g., JSOUP¹²—and determines whether the pattern appears in the page based on exact string match with stemming. This step is necessary because the search results can contain spurious pages—i.e. pages that do not contain the actual pattern. For example, a page containing the string ”Is the cell wall of a plant made of cellulose fibers?” may appear in the search result for the pattern “cell wall is made of cellulose”; but this string does not match the pattern (and hence does not support it). The crawling of the page is necessary as the snippet of the page in the result is typically retrieved from the cache, and the actual content may or may not reflect the same content.

Finally, PLATO counts the total number of pages that contain the pattern, and uses this count as the level of support for the OWL partonomy property—associated with the pattern—that could exist between the entity pair. For each entity pair, PLATO asserts the partonomy property whose associated pattern has the strongest supporting evidence, computed from the previous step. Returning to the example for the entity pair (*Cell Wall*, *Cellulose*), the supporting evidence for each pattern associated with the pair (assuming a search limit of 50) is below:

- Cell Wall is made of Cellulose, 48
- Cellulose is made of Cell Wall, 10
- Cell Wall is partly Cellulose, 50
- Cellulose is partly Cell Wall, 7

Since the pattern ‘Cell Wall is partly Cellulose’ has the strongest support, the associated property *po-stuff*—corresponding to Winston’s *Stuff-Object* relationships—is asserted, with Cellulose as the part and Cell Wall as the whole.

¹¹<http://msdn.microsoft.com/en-us/library/dd251056.aspx>

¹²<http://jsoup.org/apidocs/>

In addition to adding properties at the instance-level (i.e. between entities), PLATO also enriches the schema by generalizing from the instance level assertions. To explain this step, let C and D be two classes about which we want to find out whether they should be related on the schema level by one of the partonomic relationships R . From the process just described, a set $M_{R,C,D}$ can be obtained of instance level assertions of the form $R(a, b)$, where $a \in C$ and $b \in D$.¹³ PLATO now add schema level axioms according to the following rules: (1) If, for all $a \in C$, there is a $b \in D$ with $R(a, b) \in M_{R,C,D}$, then add the axiom $C \sqsubseteq \exists R.D$, which can be expressed in OWL/RDF serialization using the *owl:someValuesFrom* property restriction. (2) If, for all $b \in D$, there is a $a \in C$ with $R(a, b) \in M_{R,C,D}$, then add the axiom $D \sqsubseteq \exists R^-.C$, where R^- indicates the inverse (using *owl:inverseOf*) property of R . While this approach seems to be rather crude compared to schema learning methods based on inductive paradigms,¹⁴ it already achieves good results, as can be seen from the evaluation in Section 6.4.3.

6.4 Evaluation

Three experiments are presented to evaluate the performance of PLATO on enriching LOD Cloud dataset with partonomy properties. The first experiment evaluates PLATO's performance on discovering partonomy properties between entities within the same LOD Cloud dataset (i.e. intra-dataset instance-level partonomy discovery). The second experiment evaluates PLATO's performance across different LOD Cloud datasets (i.e. inter-dataset instance-level partonomy discovery). The final experiment evaluates PLATO's performance on discovery partonomy properties at the schema level. All the evaluation components of this work are available for download at the PLATO Project Page¹⁵

6.4.1 Intra-Dataset Instance-Level Partonomy Discovery

The performance of PLATO was evaluated on discovering partonomy properties between entities within the same LOD Cloud dataset using the following methodology. First, the DBpedia dataset was chosen because: 1) it is one of the largest datasets available on the Linked Open Data Cloud; and 2) it covers diverse domains

¹³If PLATO did not obtain any such assertion, then it did not add any schema axiom.

¹⁴such as [74]

¹⁵<http://wiki.knoesis.org/index.php/PLATO>

such as Geography, Science, Politics, History and Arts [18]. The scale and coverage of DBpedia allows us to thoroughly evaluate the performance of PLATO across different partonomy types [118] and domains.

Next, randomly generated 83,639 entity pairs from DBpedia are tested for evaluation because it was not practical to generate all possible entity pairs given DBpedia's size. The Mediawiki API¹⁶ is chosen to randomly generate a pair of Wikipedia articles, whose URLs were then translated to the corresponding DBpedia entities. Given that it is not practical to generate all entity pairs within DBpedia, this method provides an unbiased dataset for evaluation.

PLATO was then applied to the resulting dataset to automatically discover partonomy properties between each entity pair. For each partonomy property discovered, the property was randomly assigned to one of three human graders, who validated its correctness. A human grader determined that the partonomy property discovered by PLATO between a pair of entities is correct if the following conditions are all satisfied:

- A part-whole relationship does exist between the entities
- The correct partonomy property is given
- The part-whole roles are correctly assigned to the entities – e.g., given the pair cell and cell wall, cell is the whole and cell wall is the part.

Finally, the precision (i.e. the number of correct partonomy properties discovered by PLATO over the total number of partonomy properties discovered) based on the human grader's responses is reported. The recall for PLATO is not reported because: 1) an existing DBpedia benchmark for this purpose does not exist, and 2) the large number of entity pairs made it difficult to compute the recall manually due to time and resource limitations.

Table 6.2 shows the results for this experiment. Of the 83,639 entity pairs generated, PLATO discovered partonomy properties for 13,853 pairs. It should be noted that partonomy relationships do not exist for many of the entity pairs because these pairs were randomly generated – e.g. a random sample

¹⁶<http://en.wikipedia.org/w/api.php?action=query&list=random&rnamespace=0>

Relation Type	Distinct Entity Pairs	Correctly Found	Precision
Stuff-Object-Part-Of	4178	3427	0.82
Component-Integral-Part-Of	3126	27931	0.89
Feature-Activity-Part-Of	1287	464	0.85
Member-Collection-Part-Of	1912	803	0.85
Portion-Mass-Part-Of	0	0	NA
Place Area-Part-Of	3350	1248	0.48
Total	13853	10557	0.76

Table 6.2: Precision of the six different relation types between DBpedia entities

of 100 pairs found only 11 to have a valid partonomy relationship. PLATO was able to filter many of these extraneous pairs based on the heuristic that two entities must be strongly associated (see Section 3.1). Overall, PLATO achieved high precision in discovering partonomy properties between entities in DBpedia. Moreover, PLATO discovered partonomy properties across a wide range of entities ranging from places to chemical compounds. However, PLATO did have low precision for a couple of partonomy properties – i.e. 'Portion-Mass' and 'Place-Area'. For 'Portion-Mass', PLATO did not find any entities related to each other. This is understandable as this property deals with very abstract entities such as 'Slice of Lemon', 'Hunk of Clay', etc. and hence it's hard to find entities of this type in DBpedia.

PLATO achieved low precision for the Place-Area property because many places are ambiguous. For example, Athens can refer to either a city in Greece, Georgia, or Ohio. Similarly, Delaware can refer to either the U.S. state of Delaware or Delaware county in the U.S. state of Oklahoma. In the case of the later, given the entity pair of Delaware (State) and Oklahoma, PLATO may find false evidence supporting the hypothesis that the state of Delaware is part of Oklahoma, which can lead to poor precision. This problem can be addressed with richer partonomy semantics such as a state cannot be part of another state. These richer semantics are not captured by Winston's partonomy relationships (and hence the corresponding OWL properties), and offers a possible direction for future research.

Although recall is not reported, preliminary insights into PLATO's performance on this measure are reported. The random sample of 100 entity pairs (see above) suggests PLATO achieved good performance on this metric. Of the 11 pairs with valid partonomy properties, PLATO discovered 7 of them. Moreover, qualitative observations of sample results further suggest that PLATO performs well on recall. For example, PLATO discovered the correct partonomy property between NATO and 23 of its member states – the

total number of NATO member states is 28. Similarly, PLATO discovered the correct partonomy property between the Rock Band 'Nirvana' and all of its members – i.e. Kurt Cobain, Krist Novoselic and Dave Grohl.

6.4.2 Inter-Dataset Instance-Level Partonomy Discovery

The performance of PLATO on discovering partonomy properties is evaluated between entities from different LOD Cloud datasets using the following methodology. Two inter-dataset partonomy discovery tasks are created for: 1) discovering partonomy properties between Freebase dishes and DBpedia ingredients, and 2) discovering partonomy properties between Freebase human anatomy parts and DBpedia organs. These two tasks were chosen because:

- Freebase provides a pre-defined list of 2,615 food dishes¹⁷ and 2,916 human anatomy parts,¹⁸ which have well-defined parts (i.e. ingredient) and wholes (i.e. organ) respectively.
- DBpedia provides the corresponding parts and wholes.
- Freebase provides the ingredients for each food dish, which can be used as an independent gold standard for the first task; and experts in the medical domain were readily available to assess PLATO's performance for the second task.

PLATO was then applied to both tasks. For the Dish-Ingredient task, the partonomy properties discovered by PLATO was validated against the ingredients for each dish provided by Freebase to compute both precision (i.e. number of correct partonomy properties discovered by PLATO over all partonomy properties discovered) and recall (i.e. number of actual partonomy properties discovered by PLATO over all partonomy properties). For the Anatomy-Organ task, an independent gold standard does not exist – i.e. Freebase does not provide the organs for each anatomy part. Hence, an expert in human anatomy was employed to grade each partonomy property discovered by PLATO, and reported PLATO's precision based on the expert's response. These experts had no knowledge about PLATO and were presented the results as an exercise

¹⁷<http://www.freebase.com/view/food/views/dish>

¹⁸http://www.freebase.com/view/medicine/views/anatomical_structure

to judge if the presented ingredients are used for the given dish. The expert used the same grading criteria described in the previous experiment (see Section 4.1). The recall for PLATO is not reported because of resource and time limitations.

Task	Recall	Precision
Dish-Ingredient Task	0.72	0.53
Anatomy-Organ Task	N/A	0.86

Table 6.3: This table shows PLATO’s performance on precision and recall for the Dish-Ingredient task, and PLATO’s performance on precision for the Anatomy-Organ task. Recall was not reported for the second task because of time and resource limitations.

Table 6.3 shows the results for both tasks. For the Dish-Ingredient task, PLATO achieved high recall and modest precision. The Freebase dish gold standard consists of 2,615 dishes and a total of 1317 ingredients across these dishes. Many of the dishes do not have ingredients mentioned for them. PLATO discovered a total of 1766 partonomy relationships between Freebase dishes and DBpedia ingredients, of which 936 are valid according to the gold standard – giving a recall of 0.72 and precision of 0.53. This result demonstrates that PLATO can effectively discover partonomy properties across different LOD Cloud datasets. Interestingly, the modest precision was due to PLATO discovering additional, valid partonomy properties not present in the Freebase gold standard. For example, a stuff-object property exists between the ingredient ice cream and the dish ‘Baked Alaska’, which PLATO correctly discovered. However, the Freebase gold standard overlooked this relationship, resulting in lower precision.

Given this oversight, 2 human graders were employed to independently review each extra result generated (830 in total) to determine whether it’s due to a real erroneous result given by PLATO or a gap in the gold standard (i.e. an overlooked ingredient in a food dish). The graders used the same grading criteria described in Section 4.1 Both graders were required to agree that a response is valid in order for it to be counted as correct. The graders responses were then used to adjust the precision. They found 512 correct answers out of 830, which resulted in total correct ingredients of $936+512=1448$, an adjusted precision of 0.82 – a significant increase over the original precision.

For the Anatomy-Organ task, PLATO achieved high precision. Of the 8,397 distinct partonomy properties discovered by PLATO, the human expert verified 7,221 as correct, thus leading to a precision of 0.86. The expert in this case, is a researcher in medical science and not related to research and development of

PLATO. The expert was presented the results of PLATO as a grading exercise to judge if the assertions are right or wrong. This result further demonstrates – in a different domain – that PLATO can effectively discover partonomy properties across different LOD Cloud datasets. For example, PLATO correctly identified that the entity 'Axon' is a component-integral object part of entities such as 'dorsal root ganglion', 'synapse', 'neuron' and 'nerve'.

6.4.3 Assertion of schema level links

Using the instance level assertions which are generated between entities, it becomes possible to identify the schema level relationships, which exist between the classes of these entities, as, described at the end of Section 6.3.2. For example, using the fact that 'Nirvana has a member Kurt Cobain' and 'Queen has a member Freddie Mercury', and in fact that for all bands some member has been found which is classified as an artist, schema level assertions between DBpedia classes can be identified such as

```
dbpedia-owl:Band rdfs:subClassOf [
    rdf:type                owl:Restriction ;
    owl:onProperty        :hasMember ;
    owl:someValuesFrom    dbpedia-owl:Artist
] .
```

The schema level statement essentially says that 'Bands have members Artists'. Table 6.4 shows the evaluation of precision for schema level links, which were asserted by PLATO.

Total # of Class Pairs	Correctly Identified	Precision
93	81	0.87

Table 6.4: Precision as measured on Schema Level Links Between DBpedia entities

The entity in column 1 in Table 6.4 is the total number of distinct class pairs that were asserted to have a relationship in the file expressing schema level constraints. For example [dbpedia-owl:Artist,dbpedia-owl:Organization],[dbpedia-owl:Artist,dbpedia-owl:Artifact]. Thus, a single entity may occur in multiple

such combinations, but in each of these pairs, the entity with which it is being related to is unique. Of these 93 different pairs, a total of 81 were found to be correct, leading to a precision of 0.87. The number of class pairs found is low because many entities in the DBpedia dataset do not have any classes associated with them. Identification of schema level relationships can potentially help with improving the precision and recall of instance level relationship identification. This dataset has also been made available on the project page for download.

6.5 Related Work

To the best of our knowledge, this is the first work which, automatically identifies 'part-of' relationships in the context of the LOD cloud or RDF datasets. The field of Ontology Matching and Instance Matching has been focusing on identifying relationships such as 'sameAs', 'subClass' and 'equivalentClass.' In [38, 23] the authors present a survey in the area of ontology matching. This helps in cleaning up the data and improving the quality of links at the instance level, but the issue of identifying appropriate relationships at the schema level has not been addressed. void [1] provides a vocabulary to represent the relationships between the different datasets. SILK Framework [116] automates the process of link discovery between LOD datasets at the instance level. At the schema level, a notable effort for creating a unified reference point for LOD schemas is UMBEL [11], which is a coherent framework for ontology development and can serve as a reference framework.

There has been a number of efforts in the area of Natural Language Processing for identification of part-of relationships within a text corpora [45, 113]. This includes effort that utilizes the presence of certain lexico-syntactic patterns (Hearst patterns [56]) to indicate a particular semantic relationship between two nouns. However, much of this work has been confined to ontology learning [24] in the sense of hyponym extraction [56]. A closely related work that also mines the Web for the relations is NELL [20]. There are a few notable differences between our approach and NELL, (1) NELL uses a crawler to crawl the Web and identify relations it can find between entities on the web. PLATO is focused on LOD cloud and for a given pair of entities, PLATO tries to identify the relationship between them. (2) Predicates or properties extracted from NELL are at the surface level and do not convey the semantics of the properties. For example, while

NELL does extract facts such as Athens and Greece are related by the predicate `citycapitalofcountry`, it does not explicitly provide any semantics to those relationships. NELL has given a lot of insight and it also validates the belief that web can be mined to gain information about relationships. However, it will be extremely difficult to compare PLATO with NELL since, NELL is not available for download and systems have different set up and objectives.

The closest work in this respect is Espresso [90] that again works on a specific text corpus. A key difference of this work from PLATO is its use of a supervised approach. Further, it disregards any information about the type of entities, which can be captured using Winston's patterns.

Querying Partonomical Relationship on LOD cloud

Recently, spatial information has become widely available to consumers through a number of popular sites such as Google Maps, Yahoo Maps and Geonames.org¹. In the context of the Semantic Web, Geonames has provided RDF [83] encoding of their knowledge base. One issue that makes using the Geonames ontology, or any non-trivial spatial ontology difficult to use, is that users have to completely understand the structure of the ontology before they can write meaningful queries. To illustrate the point, consider the following query from National Geographic Bee², "In which country is the city of Pamplona? This seems to be a straightforward question, and one would assume that the logic for encoding this question into SPARQL [100] query would be to ask Return a country which contains a city called Pamplona. However, it turns out that such a simple query does not work. This is because Pamplona is a city within a state, within the country of Spain. Therefore the correct logic for encoding the question into query would be Return a country which contains a state, which contains a county, which contains a city called Pamplona. Unless the user fully understands the structure of the ontology, it is not possible to write such queries.

In this chapter, a system called PARQ (Partonomical Relationship Based Query Rewriting System) is introduced that will automatically align the gap between the constraints expressed in users query and the actual structured representation of information in the ontology. The work leverages existing work in classification of partonomic relationships [118] to re-write queries.

¹<http://geonames.org>

²<http://www.nationalgeographic.com/geobee/>

7.1 Introduction

To study the accuracy of approach for re-write, PARQ was tested on (1) 120 randomly selected questions from the National Geographic Bee and evaluated them on Geonames ontology (2) 46 randomly selected trivia questions related to British villages and counties from trivia website ³ and evaluated them on British Administrative Geography Ontology ⁴. For both the evaluations, users were instructed to read the questions and to write queries in SPARQL for the questions. PARQ rewrote the queries using partonomical relationships. The results were encouraging, and on an average, for evaluation 1, PARQ was able to re-write and answer 84 of 120 queries posed by users, whereas a SPARQL processing system could answer only 20 such queries. For evaluation 2, PARQ was able to re-write and answer 41 of 46 queries posed by users. For both the evaluations, the performance of PARQ was also compared with another well known system PPARQL [2] which extends SPARQL with path expressions to allow use of regular expressions with variables in predicate position of SPARQL.

The contributions of this work are the following:

1. This work focuses on rewriting SPARQL Queries, written from a users perspective without worrying about the underlying representation of information
2. The work utilizes partonomic transformation rules to re-write SPARQL queries
3. PARQ has been completely evaluated on third party data (queries and dataset) and shows that it is able to re-write and answer queries not answered by a SPARQL processing system. It also demonstrates PARQ can significantly improve precision without any recall loss.

7.2 Background

All spatial entities are fundamentally part of some other spatial entity. Hence, spatial query processing systems often encounter queries such as (1) querying for parts of spatial entities (for example, give me all

³<http://www.funtrivia.com/>

⁴<http://www.ordnancesurvey.co.uk/oswebsite/ontology/AdministrativeGeography/v2.0/AdministrativeGeography.rdf>

counties in Ohio) (2) querying for wholes which encompass spatial parts (for example, return a country which contains a city called Pamplona).

By identifying which relationships between spatial entities are partonomic in nature it becomes feasible to identify if queries involving those relationships fail because of part-whole mismatch and it becomes possible to fix the mismatches using transformation rules that leverage the partonomic relationships. This section provides a brief overview of work related to partonomic relationships.

This work of query rewriting to remove these mismatches is based upon using well accepted partonomic relationships to address mismatches between a users conceptualization of a domain and the actual information structure. Part/Whole relation, or partonomy, is an important fundamental relationship which manifests itself across all physical entities such as human made objects (Cup-Handle), social groups (Jury-Jurors) and conceptual entities such as time intervals (5th hour of the day). Its frequent occurrence results in manifestation of part-for-whole mismatch and whole-for-part mismatch within many domains especially spatial datasets.

Winston [118] created a categorization of part whole relations which identified and covers part whole relations from a number of domains such as artifacts, geographical entities, food and liquids. It is one of the most comprehensive categorization of partonomic relationships and other works in similar spirit such as [44] analyze his categorization.

This categorization has been created using three relational elements:

1. Functional/Non-Functional (F/NF):- Parts are in a specific spatial/temporal relation with respect to each other and to the whole to which they belong. Example: Belgium is a part of NATO partly because of its specific spatial position.
2. Homeomereous/Non-Homeomereous (H/NH):- Parts are same as each other and to the whole. Example: Slice of pie is same as other slices and the pie itself [118].
3. Separable/Inseparable (S/IN): - Parts are separable/ inseparable from the whole. Example: A card can be separated from the deck to which it belongs.

Table 6.1 illustrates these six different categories, their description using the relational elements and examples of partonomic relationships covered by them.

Using this classification and relational elements, relations between two entities can be marked as partonomic or non partonomic in nature. Further if they are partonomic, the category to which they belong is identified. Finally, appropriate transformation rules can be defined for each category to fix these mismatches.

For the purpose of this work, the focused is on the last category "Place-Area". Places are not parts of any area because of any functional contribution to the whole, and they are similar to the other places in the area as well. Also places cannot be separated from the area to which they belong. Hence, this classification can allow appropriate ontological relationships to be mapped to Place-Area category such as those found in Geonames.

7.3 Challenges

7.4 PARQ Approach

At the highest level of abstraction, PARQ takes in a SPARQL query and transforms it with the help of transformation rules. This section provides the details of the system. It describes the various modules of the system, the technologies used for building the system, the transformation rules utilized for transformation of the SPARQL queries and the motivation behind them. Finally it describes the underlying algorithm that explains how the transformation rules are utilized by PARQ for re-writing queries.

7.4.1 System Architecture

PARQ consists of following three major modules: 1) Mapping Repository 2) Transformation Rule generator and 3) Query Re-writer. Figure 7.1 illustrates the overall architecture of this system.

7.4.1.1 Mapping Repository

This module stores mappings of ontological properties to Winstons categories. These mappings are utilized by the Transformation Rule Generator to generate domain specific rules, which are consumed by the

Query Re-writer. This is the only module in the system which requires user interaction (other than for query submission). In other words, the user has to specify these mappings.

Each mapping is encoded as a rule in Jena's rule engine format where the antecedent is a triple specifying an ontological property to be mapped and the consequent is a triple specifying the Winston category that the property is mapped to. For example, the following mapping:

```
[parentFeature: (?a geo:parentFeature ?b)⇒(?a place_part_of ?b)]
```

maps parentFeature a property from the Geonames ontology to place_part_of Winstons category of Place-Area.

7.4.1.2 Transformation Rule Generator

This module automatically generates domain specific transformation rules using the mapping repository and pre-defined meta-level transformation rules based on Winstons categories of part-whole relations, which will be explained later. For example, given the following meta-level transformation rule:

```
newline [transitivity_placePartOf: place_part_of ?c] (?a place_part_of ?b)(?b place_part_of ?c)⇒(?a place_part_of ?c)]
```

newline This module will utilize the parentFeature mapping defined above to generate the following domain specific transformation rule.

```
newline [transitivity_parentFeature: (?a geo:parentFeature ?b)(?b geo:parentFeature ?c)⇒(?a geo:parentFeature ?c)]
```

newline The resulting rule is used by the Query Re-writer to re-write the graph pattern of SPARQL queries in the event of a partonomic mismatch.

This design enables PARQ to be easily used with a wide-range of ontologies. The knowledge engineer only needs to specify the mappings between properties of these ontologies and Winstons categories, which requires less effort than generating the domain-specific transformation rules themselves. This design also allows the transformation rules to be extended in an ontology agnostic manner.

This module has been implemented using Jena's ⁵ rule engine API. Like the mappings, the meta-level

⁵<http://openjena.org>

transformation rules and the generated rules are encoded in the format accepted by Jena rule engine API. The rule engine allows reading, parsing and processing of rules along with the creation and serialization of new rules.

7.4.1.3 Query Re-writer

. This module re-writes a SPARQL query in case of a partonomic mismatch between the query and the knowledge base to which the query is posed. This module is implemented using Jena and ARQ API ⁶. Jena and ARQ provide functionality to convert a query into algebraic representation and vice versa. The triples specified in the query are identified. If they map to partonomic relation using the mapping repository and using Jena's Rule Engine API, the domain specific transformation rule, appropriate transformation is performed on the triples. These transformations are then utilized to re-write the triples exhibiting the mismatch using the features provided by ARQ API.

We believe including transitivity as a part of the reasoner can result in significant overhead for large datasets such as geonames where transitivity applies to almost all the entities. By including it as a part of query rewriting method (1) it allows the mismatches to be resolved on an "on demand" basis (2) it makes it easy to plug in support for resolving other kinds of mismatches.

7.4.2 Meta-level Transformation Rules

Meta-level transformation rules are used to generate domain-specific rules that are used to resolve mismatches resulting from differences in encoding between the granularity of query constraints and the knowledge base by transforming the encoding of the constraints in the query to match the knowledge base.

These meta-level rules are defined at the level of Winstons categories, and a rule defined for a particular category applies to only the partonomic relations covered by that category. For example, rules defined for Component-Object category will cover only relations between machines and their parts, organization and their members, etc.

⁶<http://openjena.org>

The following methodology was used to define the meta-level rules used by our system. First, previous work by Varzi[21, 114] and Winston was leveraged, who both showed the semantics of transitivity holds true as long as it is applied across the same category of partonomic relation. From this result, we defined the meta-level transitive transformation rules shown in Table 2, that correspond to Winstons six part-whole categories.

Next, the interaction between Winstons categories was investigated by examining all possible combinations of these categories for additional transformation rules. This investigation, however, resulted in only frivolous rules, which were not useful for resolving mismatches. For example, the following transformation rule resulted from composing the Feature-Activity category with the Place-Area category.

(a place_part_of b) (b feature_part_of c) $\Rightarrow$ (a feature_part_of c)

However given the following query and triples in an ontology (given in English for brevity),

QUERY: "What state was attacked in WW-II?"

TRIPLE 1 : Florida is a place part of USA (Place-Area). TRIPLE 2: USA was attacked in WW-II (Feature-Activity)

The rule incorrectly transformed this query to match the ontology, that resulted in an incorrect answer being returned (i.e. Florida).

The reason for these frivolous rules is because Winstons categories are mutually exclusive as they are defined using relational elements. Hence, the meta-level transformations consist of only transitive rules. Despite this small number of rules, it was found through evaluation that transitivity by itself provide significant leverage in resolving part-whole mismatches.

7.4.3 Algorithm

The algorithm used in applying transitivity for resolving mismatches is show in Fig 7.4.3

The intermediate nodes are replaced such that the object and subject of contiguous triples have the variable names. Replace the triple in the graph pattern with the path containing the variables.

7.4.3.1 Explanation

Let us explain the algorithm using a query In which county can you find the village of Crook that is full of lakes? If the SPARQL Query submitted by user for this question is

```
SELECT ?countyName
WHERE
{
  ?village ord:hasVernacularName "Crook" .
  ?county rdf:typeord:County ;
          ord:hasVernacularName ?countyName ;
          ord:spatiallyContains ?village .
}
```

Step 1: The system compiles the query to verify if it is well formed. Since, in this case it is a well written query, the system moves on to Step 2.

Step 2: Step 2: The query is converted into its algebraic representation, and the system iterates through its list of triples to identify triples containing partonomic relationship using the mapping file provided by the user. In this case the last triple

`t=?county ord:spatiallyContains ?village`

contains spatiallyContains property which indicates that the object is part of the subject. Hence, this triple is identified as a triple for re-writing.

step 3: The other triples which contain the variables mentioned in t,such as:

```
?village ord:hasVernacularName "Crook".,
?county rdf:typeord:County.
?county ord:hasVernacularName ?countyName.
```

are utilized for unifying the values of variables of t (i.e. ?village and ?county). Using these ?village = `osr7000000000013015` which is the resource for Crook in Administrative Geography Ontology and ?county=set of resources belonging to counties is computed.

Step 4: The set of unified values from Step3 is then utilized to compute a path by executing transformation rule of transitivity involving the property `tangentiallySpatiallyContains`, `completelySpatiallyContains` `?place =osr7000000000013015` following path being returned: `?county=List of counties`. This results in the

1. `osr7000000000013244 tangentiallySpatiallyContains osr7000000000012934`
2. `osr7000000000012934 completelySpatiallyContains osr7000000000013015`

Step 5: In the path, the source and destination are replaced as mentioned in the original query, and the intermediate node is consistently replaced by a variable.

1. `?county ord:tangentiallySpatiallyContains ?var`
2. `?var ord:completelySpatiallyContains ?village.`

Step 6: In the original query the last triple is replaced by these two triples resulting in the following query

```
SELECT ?countyName
WHERE
{
    ?village ord:hasVernacularName "Crook" .
    ?county rdf:type ord:County ;
        ord:hasVernacularName ?countyName ;
        ord:tangentiallySpatiallyContains ?var .
    ?varord:completelySpatiallyContains ?village .
}
```

There can be certain cases where a number of paths are computed between two end points because of transitivity. This will result in generation of multiple re-written queries. The generated queries are ranked using the following parameters: (1) Re-written queries generating results are given higher ranking than ones which do not (2) If both queries generate results, in those scenarios queries requiring minimum amount of re-writing are given a higher ranking.

7.5 Evaluation

The objective is to determine whether the approach enables users to successfully pose queries about partonomic information to ontology where the users are not familiar with its structure and organization. This lack of familiarity will result in many mismatches that need to be resolved in order to achieve good performance.

To evaluate the objective, Geonames⁷ and British Ordinance Survey Administrative Geography Ontology⁸ were chosen as ontologies because: (1) they are one of the richest sources of partonomic information available to the semantic web community. (2) they are rich in spatial information. Geonames has over 8 million place names such as countries, monument, cities, etc. which are related to each other via partonomic relationships corresponding to Winstons category of Place-Area. For example, cities are parts of provinces and provinces are parts of countries. Table 7.1 shows some key relationships found in Geonames.

property	description
name	Name of the place
featureCode	Identifies if the place is a country, city, capital etc.
parentFeature	Identifies that the place identified by domain is located within the place identified by the range

Table 7.1: Important Properties in Geonames

Similarly, Administrative Geography Ontology provides data related to location of villages, counties and cities of the United Kingdom which again map to Winstons place-area relation. Table 7.2 shows the description of key administrative geography ontology properties. Namespace has been omitted for brevity.

For evaluating our approach on Geonames ontology, a corpus of queries was constructed for evaluation by randomly selecting 120 questions from previous editions of National Geographic Bee⁹, an annual competition organized by the National Geographic Society which tests students from across the world on their knowledge of world geography. For British Administrative Geography ontology, 46 questions were selected from a popular trivia website¹⁰ that hosts a number of quizzes related to British geography. These questions were chosen for evaluation because:

⁷<http://geonames.org>

⁸<http://www.ordnancesurvey.co.uk/oswebsiteontology/AdministrativeGeography/v2.0/AdministrativeGeography.rdf>

⁹<http://www.nationalgeographic.com/geobee/>

¹⁰<http://www.funtrivia.com/>

property	description
spatiallyContains	The interior and boundary of one region is completely contained in the interior of the other region, or the interior of one region is completely contained in the interior or the boundary of the other region and their boundaries intersect.
tangentiallySpatiallyContains	The interior of one region is completely contained in the interior or the boundary of the other region and their boundaries intersect. It is a subproperty of spatiallyContains.
completelySpatiallyContains	The interior and boundary of one region is completely contained in the interior of the other region. It is a sub-property of spatiallyContains.

Table 7.2: Important Properties in Administrative Geography Ontology

1. These questions are publicly available, so others can replicate the evaluation.
2. Each question has a well-defined answer, which avoids ambiguity when grading the performance of this approach.
3. These questions are of places and their partonomic relationship to each other. Hence, there is significant overlap with Geonames and Administrative Geography Ontology.

Examples of such questions include:

- The Gobi Desert is the main physical feature in the southern half of a country also known as the homeland of Genghis Khan. Name this country.
- In which English county, also known as "The Jurassic Coast" because of the many fossils to be found there, will you find the village of Beer Hackett?

Once the questions were selected, 4 human respondents were employed (computer science students at a local university) to encode the corresponding SPARQL query for each question. These respondents are familiar with SPARQL (familiarity ranged from intermediate to advanced) but are not familiar with Geonames or Administrative Geography Ontology. These two conditions meet the evaluation objective.

For the National Geographic Bee questions, each subject was given all 120 questions along with a description of the properties in the Geonames ontology. Each subject was then instructed to encode the SPARQL query for each question using these properties and classes.

For the trivia questions, only one human respondent was employed to encode the corresponding SPARQL query because of limitations in time and resources. This respondent was given all 46 questions along with a description of the properties in the administrative geography ontology.

These instructions, original queries, responses and source code is available for download at <http://knoesis.wright.edu/stu>

7.5.1 Geonames Results and Discussion

The approach was compared to PPARQL and SPARQL. PPARQL [2] extends SPARQL with path expressions to allow use of regular expressions with variables in predicate position of SPARQL. The regular expression patterns allowed in PPARQL grammar can be constructed over the set of uris, blank nodes and variables. For example, the following query when posed to PPARQL returns all cities connected to the capital of France by a plane or train.

```
Select ?City2
WHERE
{ ?City1 ex:capital ex:France .
  ?City1 (ex:plane | ex:train) ?City2 . }
```

Queries encoded by human respondents (see previous subsection) were posed to SPARQL and PARQ. The performance of each approach was graded using the metrics of precision (i.e. the number of correct answers over the total number of answers given by an approach) and recall (i.e. the number of correct answers over the total number of answers for the queries). An approach was assumed to have correctly answered a query if its answer was the same as the answer provided by the National Geographic Bee.

Figure 7.2 shows the result of this evaluation for PARQ and SPARQL. PARQ on an average correctly re-writes 84 queries of the 120 posed by users performing significantly better than SPARQL processing system across all respondents ($p < 0.01$ for the X2 test in each case). The low performance (61 queries by using PARQ and 19 by SPARQL) for respondent 3 can be attributed to this subject having the least familiarity with writing queries in SPARQL and writing improper SPARQL queries. The high performance (103 queries using PARQ and 33 using SPARQL) for respondent 4, can be attributed to this subject having

the most experience with SPARQL. For each respondent, the difference of 120 and re-written queries is the number of queries not re-written using PARQ.

Figure 7.1 depicts the workflow, which is described in more detail in the subsequent sections.

For this comparison, the execution time of PARQ to PPARQL was also compared as shown in Figure 7.3. Because of limitations in time and resources, only one respondent was asked to encode the queries posed to PPARQL. Hence, Respondent 4 was selected because this respondent has the most experience and familiarity with SPARQL.

Although PARQ and PPARQL deliver the same recall (86.7%), the results clearly illustrate that PARQ performs much better than PPARQL in precision ($p < 0.01$ for X2 test) because of retrieval of multiple answers by PPARQL even when the particular resource was present only once in the ontology, thus exhibiting a flaw in the underlying algorithm or implementation. It also illustrates that PPARQL takes almost 95% more time on, average in answering a query than PARQ ($p < 0.05$ for 2-tailed pair-wise t-test).

These results show that mismatches are common when posing queries to an ontology and that this approach can successfully resolve these mismatches which enabled more queries to be correctly answered.

For example, given the question:

In which country is Grand Erg Oriental?

Most of the subjects produced the following query.

```
PREFIX geo:<http://www.geonames.org/ontology\#>
SELECT ?countryname
WHERE
{
  ?country geo:featureCode geo:A.PCLI.
           geo:name ?countryname.
  ?placegeo:name "Grand Erg Oriental";
           geo:parentFeature ?country.}
```

This query, however, failed to return any results when posed to Geonames because in Geonames Grand Erg Oriental is represented as a part of Tunis al Janubiyah Wilayat (a state) which is a part of Tunisia (a

country). PARQ was able to rewrite the original query to align with Geonames (see rewritten query below) which enabled the correct result to be retrieved (i.e. Tunisia).

```
PREFIX geo:<http://www.geonames.org/ontology\#>
SELECT ?countryname
WHERE
{
    ?country geo:featureCode geo:A.PCLI;
            geo:name ?countryname.
    ?place geo:name "Grand Erg Oriental".
            geo:parentFeature ?var.
    ?var geo:parentFeature ?country.
}
```

7.5.2 Administrative Geography Ontology Results and Discussion

For the questions related to British villages and counties, the results were also compared our approach to PSPARQL. PARQ was not compared to SPARQL because it delivered poor performance in the previous evaluation. Because of time and resource limitations, only one respondent was asked to serialize trivia questions related to British Villages for PARQ and PSPARQL. Again, Respondent 4 was selected for this task because this respondent has the most experience and familiarity with SPARQL, The performance of each approach was graded using precision and recall, and the execution time of both approaches was also compared. An approach was assumed correctly answered a query if its answer was the same as the answer provided by the trivia website. As illustrated in Figure 7.4 PSPARQL and PARQ perform equally well for recall, but PARQ has a much better precision than PSPARQL ($p < 0.01$ for X2 test). It also illustrates PSPARQL on an average is 28 times slower than PARQ ($p < 0.05$ for the 2-tailed pair-wise t-test).

These results again illustrate the fact that part-for-whole and whole-for-part mismatches are common in spatial ontologies and PARQ helps resolve these mismatches allowing users to write queries without worrying about the structure of the ontology. As for example for the following trivia question In which

English county, also known as "The Jurassic Coast" because of the many fossils to be found there, will you find the village of Beer Hackett?.

The user poses the following SPARQL query for the question (Namespace omitted for brevity).

```
SELECT ?countyName
WHERE
{ ?village ord:hasVernacularName "Beer Hackett" .
  ?county rdf:type ord:County ;
           ord:hasVernacularName ?countyName ;
           ord:spatiallyContains ?village .
}
```

The above specified query will not fetch any results because (1) the instance data for Administrative Geography models information using two subproperties of spatiallyContains namely tangentiallySpatiallyContains and completelySpatiallyContains. (2) Villages may or may not be directly part of counties and may contain additional administrative divisions in between.

Unfortunately the difference between tangentiallySpatiallyContains and completelySpatiallyContains is very subtle and makes it extremely difficult for a nave user to correctly identify and use the property for querying the ontology, unless the user looks at the instance data and identifies the properties. However, the property spatiallyContains is a parent property of both tangentiallySpatiallyContains and completelySpatiallyContains and is perhaps the most intuitive property of the ontology which captures the semantics of both the properties and can be used by a user for posing queries. So when the above mentioned query is re-written by PARQ according to ontology as following, it retrieves the correct result of Dorset.

```
SELECT ?countyName
WHERE
{ ?village ord:hasVernacularName "Beer Hackett" .
  ?county rdf:type ord:County ;
           ord:hasVernacularName ?countyName ;
```

```
ord:tangentiallySpatiallyContains ?var .  
?var ord:completelySpatiallyContains ?village .  
}
```

7.5.3 Summary of Results and Limitations

Based on our experiments performed it has been demonstrated that PARQ significantly improves precision without any loss in recall and performs significantly faster as well over other systems. Although our approach significantly improved performance over PSPARQL and SPARQL, there were several queries that it could not answer. Our analysis uncovered the following reasons:

- Several queries (e.g. those about political entities) could not be answered because of insufficient information in Geonames. Example of such queries includes The Cayman Islands are a territory of which country?
- Some queries required additional transformations beyond the ones identified by this work. These transformations involve relations such as containment and overlap of entities which cannot be defined in terms of Winstons categories. Hence, there is a need to extend Winstons categories to handle these types of mismatches. Example of such queries includes Which continent contains the largest number of landlocked countries?
- Some questions required features, such as aggregate functions, that are not part of the standard SPARQL specification. The focus of this work is to provide support for features which are part of standard SPARQL specification. Example of such queries includes Not including Taiwan, how many provinces comprise China?

7.6 Related Work

To the best of our knowledge this is the first work which tries to allow users to formulate SPARQL queries from their perspective without having to worry about the structure of the ontology. However, there are

existing works related to RDF Query processing and retrieval of spatial information some of which are worth mentioning to highlight their salient features and distinguish this work from them.

The use of Semantic Web technologies for better retrieval of spatial information by incorporating data semantics and exploiting it during the search process was illustrated in [35]. Building upon the vision of [35], for retrieval of spatial information, in a previous work [92] operators have been defined to query spatial, temporal and thematic information from RDF datasets. PARQ's approach for retrieval of spatial information in that work utilizes metric parameters such as geometric co-ordinates, radius, buffer for defining various operators. The operators enhance the standard spatial operators provided by Oracle Spatial and are implemented as supplemental to SPARQL. The reliance on metric parameters compliments this approach here which relies on utilization of named relationships.

Another interesting approach for querying spatial information using SPARQL [70] advocates re-modeling of ontology, than extending SPARQL for retrieval of information. Because of the emphasis on remodeling ontology than transformation of query, this work is obviously along a different dimension than our work. But the work discusses shortcomings of SPARQL for querying spatial data and discusses some interesting query types which a language tailored for spatial querying should be able to handle and hence motivates this work. In [71] authors discuss a system for storing spatial and semantic web data efficiently without sacrificing query efficiency which in future can help in supporting various other kinds of queries.

In [3] have been defined operators for identifying paths in RDF dataset given a source and destination. Using these operators it is possible to express constraints such as the length of the path, specifying a particular node to include in the paths etc. This work differs from these works since this work is not on identifying paths. Additionally, PARQ re-writes SPARQL queries and does not require specification of source and destinations for results to be retrieved. In [95, 2] investigate incorporation of regular expressions in the predicate position of SPARQL queries. Though some of these works can be used for answering the queries they suffer from issues of poor precision and slower execution time as demonstrated through the evaluation. Query re-writing has been investigated in other research areas such as databases for yielding better execution plans, data integration and semantic data caching in client-server system [51]. In context of query languages for structured graph data models, [30, 29] deal with queries that involve transitive or repetitive patterns of relations in context of databases.

There has been work in spatial query processing system for retrieval of information using partonomic relation such as in [22, 115], but not in the context of SPARQL and not utilizing named relationships. These works rely on the use of metric relations such as radius, distance etc. [22] focus on creation of composite or higher order objects via the process of thematic and spatial abstraction.

The work which comes close to this approach is [69]. The work utilizes OWL-DL entailment rules for re-writing SPARQL to retrieve inference results. Unlike this approach where the original graph pattern is altered, the queries are altered by extending graph pattern using UNION construct of SPARQL. In the absence of an accessible implementation, it is difficult to compare this approach with the system.

Another work SPARQL-DL [104] incorporates the semantics of SPARQL in their DL reasoner and hence, is along a different dimension than this work. Some other works on query rewriting are related to Query Optimization [54], but in this work the concern is with retrieval of information from spatial datasets by harnessing partonomic relationships than its optimization.

7.7 Conclusion

In this chapter an approach for supporting SPARQL rewriting to allow users to write queries from their perspective without having to worry about the structure of the ontology has been presented. The experiments have been completely performed on third party dataset and queries. Using the experimental results it can be proven that the system re-writes these queries using transformation rules such as transitivity effectively and thus helps in resolving the mismatch between query constraints and underlying knowledge base while maintaining a high level of precision of results. Further it can be demonstrated that PARQ is significantly faster and can improve precision without any loss to recall.

Figure 7.1: PARQ system flow chart

SPR= Set of Partonomic Relation
 If the query is not well formed
 return
 else
 Convert the query Q into its algebraic representation (AR).
 Identify the graph pattern(GP) and query variables(QV).
 For every triple $t \in GP$
 if $t.property \in SPR$
 If $t.subject$ is a variable
 Identify other triples with $t.subject$ and use them to unify $t.subject$
 Insert unified values in $s.List$
 else
 Insert $t.subject$ in $s.List$
 If $t.object$ is a variable
 Identify other triples with $t.object$ and use them to unify $t.object$
 Insert unified values in $o.List$
 else
 Insert $t.object$ in $o.List$
 for each $s \in s.List$
 for each $o \in o.List$
 path= Find path between s and o using the transformation rule.
 If (path! =null)
 Replace the resources in the path such that,
 path.source = $t.subject$.
 path.destination = $t.object$
 The intermediate nodes are replaced such that the object and subject of contiguous
 triples have the variable names.
 Replace the triple in the graph pattern with the path containing the variables.
 Return the query Q' to the user

	System	# of queries answered	Precision	Recall
Respondent1	PARQ	82	100%	68.3%
	SPARQL	25	100%	20.83%
Respondent2	PARQ	93	100%	77.5%
	SPARQL	26	100%	21.6%
Respondent3	PARQ	61	100%	50.83%
	SPARQL	19	100%	15.83%
Respondent4	PARQ	103	100%	85.83%
	SPARQL	33	100%	27.5%

Figure 7.2: PARQ Results on Geonames

System	Precision	Recall	Execution time/query in seconds
PARQ	100%	86.7%	0.3976
PSPARQL	6.414%	86.7%	37.59

Figure 7.3: Comparison PSPARQL and PARQ on Geonames for respondent 4

System	Precision	Recall	Execution time/query in seconds
PARQ	100%	89.13%	0.099
PSPARQL	65.079%	89.13%	2.79

Figure 7.4: Comparison for Ordnance Survey Dataset for Respondent 4

LOQUS: Linked Open Data SPARQL Querying System

8.1 Introduction

The Linked Open Data (LOD) methodology has recently emerged as a powerful way of linking together disparate data sources [17]. Using this methodology, researchers have interlinked data from diverse areas such as life sciences, nature, geography, and entertainment. Moreover, many prominent datasources (e.g. Wikipedia¹, PubMed², data.gov³, etc.) – have also adopted this methodology to interlink their data.

The result is the LOD cloud⁴ – a large and growing collection of interlinked public datasets represented using RDF and OWL. Concepts (and instances) in a dataset are connected to (and hence can be reached from) related concepts (and instances) from other datasets through semantic relationships such as *owl:sameAs*. Hence, the LOD cloud is becoming the largest currently available structured knowledge-base. It has a potential for applicability in many AI-related task such as open domain question answering, knowledge discovery, and the Semantic Web.

An important prerequisite before the LOD cloud can enable these goals is allowing its users (and applications) to effectively pose queries to and retrieve answers from it. This prerequisite, however, is still an open problem for the LOD cloud. For example, in order to answer the following query from Jamendo⁵

¹http://en.wikipedia.org/wiki/Main_Page

²<http://www.ncbi.nlm.nih.gov/pubmed/>

³<http://data.gov>

⁴<http://linkeddata.org/>

⁵<http://dbtune.org/jamendo/>

using the LOD cloud:

Select artists within Jamendo who made at least one album tagged as ‘punk’ by a Jamendo user, sorted by the number of inhabitants of the places they are based near.

This query requires user to select the relevant datasets, identify the concepts in these datasets that the query maps to, and merge the results from each dataset into a complete answer. These steps are very costly in terms of time and required expertise which is not feasible given the size (and continued growth) of the LOD cloud. Apart from the sheer size, issues such as schema heterogeneity and entity disambiguation identified in [63] present profound challenges with respect to querying of the LOD cloud.

In this paper, we present a Linked Open Data SPARQL Querying System (LOQUS) – which allows users to effectively pose queries to the LOD cloud without having to know the exact structure and links between its many datasets. LOQUS automatically maps the user’s query to the relevant datasets (and concepts) using an upper level ontology; then executes the resulting query; and finally merges the results into a single, complete answer.

We perform a qualitative evaluation of LOQUS on several real-world queries and demonstrate that LOQUS allows users to effectively execute queries over the LOD cloud without a deep understanding of its datasets. We also compare LOQUS with existing query systems for the LOD cloud to highlight the pros and cons of each approach.

The rest of the paper is organized as follows. We begin by providing the motivation behind our work. We then introduce our approach followed by an end-to-end example and evaluation. We conclude with related work, conclusion, and future work.

8.2 Motivation

SPARQL⁶ has emerged as the de-facto query language for the Semantic Web community. It provides a mechanism to express constraints and facts, and the entities matching those constraints are returned to the user. However, the syntax of SPARQL requires users to specify the precise details of the structure of

⁶<http://www.w3.org/TR/rdf-sparql-query/>

the graph being queried in the triple pattern. To ease querying from an infrastructural perspective, data contributors have provided public SPARQL endpoints to query the LOD cloud datasets. But with respect to a systematic querying of the LOD cloud, we believe that the following challenges identified previously in [63] make the process difficult and should be addressed.

- *Intimate knowledge of datasets:* To formulate a query which spans multiple datasets (such as the one mentioned in the introduction) the user has to be familiar with multiple datasets. The user also has to express the precise relationships between concepts in the RDF triple pattern, which even in trivial scenarios implies browsing at least two to three datasets.
- *Schema heterogeneity:* The LOD cloud datasets cater to different domains, and thus require different modeling schemes. For example, *a user interested in music related information has to skim through at least three different music related datasets such as Jamendo, MusicBrainz, MySpace.* Even though the datasets belong to same domain, each have been modelled differently depending on the creator. This is perfectly fine from a knowledge engineering perspective, but it makes the querying of the cloud difficult as it requires users to understand the various heterogeneous schemas. This issue stems from the *Lack of Conceptual Description of the LOD datasets.*
- *Entity disambiguation:* Often the LOD cloud datasets have overlapping domains and tend to provide information about the same entity. To exemplify, both DBpedia and Geonames have information about the city of Barcelona. Although Geonames references DBpedia using the `owl:sameAs` property, which can confuse the user as to which is the best source to answer the query. This problem gets even more compounded when contradictory facts are reported for the same entity by different datasets. *For example, DBpedia quotes the population of Barcelona as 1,615,908, whereas according to Geonames it is 1,581,595.* One can argue this might be because of a difference in the notion of *the city of Barcelona.* But that leads to another interesting question: *Is the owl:sameAs property misused in the LOD cloud?*
- *Ranking of results:* In scenarios where the results of the query can be computed and returned by multiple datasets, the result which should be ranked higher for a specific query becomes an interesting and important question. As presented above, the query related to *population of Barcelona* can be

answered by multiple datasets, but which one of them is more relevant in a specific scenario?. This issue has been addressed from the perspective of popularity of datasets by considering the cardinalities and types of the relationships in [110], but not from the perspective of requirements with regard to a specific query.

8.3 Our Approach

From a bird's eyes perspective, LOQUS accepts SPARQL queries serialized by the user using concepts from an upper level ontology. LOQUS identifies the datasets and the corresponding queries to be executed on these datasets using primarily the mappings of upper level ontology to these LOD cloud datasets. This section introduces the architecture of our querying system, approach used for query execution, and the utilization of mappings for sub-query construction and the technique used for processing the results. Figure 8.1 illustrates the overall architecture of LOQUS.

8.3.0.0.1 System Architecture LOQUS consists of the following modules (1) Upper level ontology mapped to the domain specific LOD datasets. (2) Module to identify the upper level concepts contained in the query and perform the translations to the LOD cloud datasets. (3) Module to split the query mapped to LOD datasets concepts into subqueries corresponding to different datasets. (4) Module to execute the queries remotely and process the results and deliver the final result to the user.

8.3.0.0.2 Upper Level Ontology The upper level ontology has been created manually by reusing concepts from SUMO [88] and by identifying their equivalent or subsuming concepts in the LOD cloud datasets. To demonstrate, the SUMO concept of *Nation* can map to different concepts belonging to the datasets of the LOD cloud such as <http://dbpedia.org/ontology/Country> (DBpedia), <http://www.geonames.org/ontology#A.PC> (Geonames) and <http://data.linkedmdb.org/resource/movie/country> (linkedmdb). These mappings are at the schema level, and thus complement the existing mappings at the instance level provided by LOD cloud. Thus, reusing SUMO provides a single point of reference for querying the LOD cloud and consequently helps in query formulation. Further, because the mappings are at the schema level, the ontology can be utilized for reasoning and knowledge discovery over LOD cloud datasets.

Figure 8.1: LOQUS Architecture

8.3.0.0.3 Mapping of Upper Level Concepts to LOD Datasets Using the mappings from SUMO, the concepts specified in the query can be mapped to concepts of the LOD cloud datasets. The concepts from LOD cloud dataset are substituted in the basic graph pattern (in lieu of concepts from SUMO) of the SPARQL query to create a query containing only concepts from the LOD datasets. The presence or absence of multiple mappings for a given concept gives an indication if the corresponding subqueries (which are created in the next step) should be involved in a *union* or if they should be *joined* to each other. Hence, this step also helps in creating a query plan for the execution and processing of results of the sub-queries.

8.3.0.0.4 Splitting of the Query Graph to Create Sub-Queries The SPARQL query containing the concepts from the LOD cloud datasets is partitioned into sub-queries corresponding to the datasets whose concepts are being used in the query. The division of the original query graph is done by analyzing the namespaces of the concepts and taking cognizance of the fact, that some vocabularies such as FOAF and

SIOC are reused by other datasets.

8.3.0.0.5 Execution of Queries and Processing of Results The foundation of the LOD cloud is on the reuse of URIs across datasets typically to assert similarity between concepts or to link them. In order to search for concepts similar to the variables of the queries created in the previous step, their graph is appended with triples querying for "owl:sameAs", "skos:closeMatch" and similar relations using the OPTIONAL pattern of SPARQL. This step helps in identifying similar concepts and also join results from different datasets.

The results retrieved from the execution of the queries are processed according to the query plan. For example, assume that the query plan suggests that results for execution of query "Search for nations and their corresponding populations" (executed on Geonames and DBpedia), should be in a "union" with each other. To perform this operation similar concepts are identified and grouped together. The similarity is identified by using *similarity properties* such as "owl:sameAs" or "skos:closeMatch". Thus, the Geonames resource for Haiti <http://sws.geonames.org/3723988/> can be linked to the CIA Factbook concept <http://www4.wiwiwiss.fu-berlin.de/factbook/resource/Haiti> by using the equivalence established by the DBpedia concept for Haiti <http://dbpedia.org/page/Haiti>, using an "owl:sameAs" link. Hence, answers from sub-queries can be merged and joined together. This mechanism also allows for finding results in scenarios which do not have a direct link by traversing some common well known similarity properties as mentioned above and retrieving information from there.

8.3.0.0.6 Scenario Illustration A query submitted by the user using the upper level ontology searching for "*Identify films, the nations where they were shot and the population of these countries*" undergoes the following process

1. The user looks at the upper level ontology to identify the relevant concepts and serializes them into a SPARQL query.

```
Select ?film ?nation ?pop
WHERE
{ ?film sumo:location ?nation;
  rdf:type sumo:film.
```

```
?nation rdf:type sumo:Nation;
      sumo:population ?pop.}
```

2. By utilizing the mappings the LOD cloud dataset specific query concepts are substituted in lieu of upper level ontology concepts.

```
Select ?film ?nation ?pop
WHERE {?film linkedmdb:country ?nation;
      rdf:type linkedmdb:film.
      ?nation rdf:type dbpedia:Country;
      geo:featureCode geo:A.PCLI;
      dbprop:populationCensus ?pop ;
      geo:population ?pop.}
```

3. By identifying the different datasets to which the concepts mentioned in the query graph pattern belongs, various sub-queries are created (each of which belong to a separate dataset). The query plan is also generated at this step by identifying if upper level ontology concept has multiple mappings or single mapping to LOD cloud dataset. For example, results of queries executed on datasets which provide demographic information such as DBpedia and geonames will be in "UNION", whereas LinkedMDB query results would be joined with these results.

```
{

SELECT ?nation ?pop ?nation1 ?propertyvar
WHERE {?nation  rdf:type db:Country;
      dbprop:populationCensus  ?pop.
OPTIONAL{?nation owl:sameAs ?nation1.}
OPTIONAL{?nation skos:closeMatch ?nation1.}
OPTIONAL{?nation ?propertyvar ?nation1.}

}
```

UNION

```
SELECT ?nation ?pop ?nation1 ?propertyvar
WHERE {?nation geo:featureCode geo:A.PCLI;
      geo:population  ?pop.
OPTIONAL{?nation owl:sameAs ?nation1.}
OPTIONAL{?nation skos:closeMatch ?nation1.}
OPTIONAL{?nation ?propertyvar ?nation1.}

}
```

```
}

```

JOIN

```
SELECT ?Film ?nation ?nation1 ?propertyvar
WHERE {?Film linkedmdb:country ?nation;
      rdf:type linkedmdb:film.
      OPTIONAL{?nation owl:sameAs ?nation1.}
      OPTIONAL{?nation skos:closeMatch ?nation1.}
      OPTIONAL{?nation ?propertyvar ?nation1.}
}
```

4. Using an available mapping of datasets and their corresponding SPARQL endpoints, the sub-queries are executed and the Table 8.1 to Table 8.3 illustrates some of the results fetched by the three sub-queries given above.

Nation	Nation1	Population	Property Var
geo:102358	db:Saudi_Arabia	28161000	owl:sameAs
geo:1036973	db:Mozambique	21284000	owl:sameAs
geo:1269750	db:India	1147995000	owl:sameAs

Table 8.1: Result execution of queries over geonames

5. Finally the results of these sub-queries are processed according to the preidentified query plan. The results to be involved in UNION are merged using equivalence properties such as "owl:sameAs", whereas the query results to be in JOIN are combined by looking for similar concepts. The generated results as illustrated in Table 8.4 are returned to the user.

8.4 Evaluation

As a proof of concept we have implemented LOQUS using the Jena⁷ Semantic Web Framework. The system takes a SPARQL query serialized by the user using concepts from the upper level ontology, and performs

⁷<http://jena.sourceforge.net/>

Nation	Nation1	Population	Property Var
db:Saudi_Arabia	geo:102358	28,686,633	owl:sameAs
db:Saudi_Arabia	cyc:en/SaudiArabia	28,686,633	owl:sameAs
db:Mozambique	cyc:en/Mozambique	21,397,000	owl:sameAs
db:Mozambique	umbel:Mozambique	21,397,000	owl:sameAs

Table 8.2: Result execution of queries over dbpedia

Film	Nation	Nation1	Property Var
lmdb:30356	lmdb:IN	geonames:1269750	skos:closeMatch
lmdb:27302	lmdb:SA	geonames:102358	skos:closeMatch
lmdb:35434	lmdb:MZ	geonames:1036973	skos:closeMatch

Table 8.3: Result execution of queries over linkedmdb

Film	Nation	Population	Nation	Population
lmdb:30356	geo:1269750	1147995000	db:India	1028610328
lmdb:27302	geo:102358	28161000	db:Saudi_Arabia	28,686,633

Table 8.4: Result of user submitted query

the appropriate mapping. LOQUS then executes the query and merges the results and presents the results to the user.

We perform a qualitative evaluation of our system with DARQ [96] and SQUIN [53]. Our objective is to determine whether our system allows users to execute and retrieve answers to SPARQL queries over the LOD cloud without knowing the individual datasets and by just using the concepts from the upper level ontology. The lack of specification of LOD datasets in the queries requires good quality mappings to correctly identify the datasets which can be useful in answering the queries. Further, the system has to provide an efficient processing of the results for combining the results of sub-queries.

A standard measure for assessing the quality of querying systems are precision and recall. In our case, however, there does not exist any benchmarks or even available baselines for measuring these statistics partly because this is an emerging area. The sheer size of the LOD cloud makes it difficult to identify if all correct answers have been retrieved and reported. Currently there is no easy way to create a baseline for a large set of LOD cloud queries because there are no available systems which can perform the task in a complete manner, as required for creating a baseline reference. At the same time, SPARQL endpoints also restrict the number of results returned for a specific query. Hence, getting complete sets of answers is a challenge.

8.4.0.0.7 Queries and Results To evaluate our objective we took queries which require information from multiple LOD datasets and serialized them into SPARQL queries using concepts from the upper level ontology. Table 8.5 presents some of the queries used for evaluating LOQUS along with statistics related to the execution of these queries. The queries though small in number require information from different sections of the LOD cloud and some of them have been adopted from publicly available sources. The queries have been executed successfully by LOQUS in a manner similar to Query 1 (which is explained

in **Scenario Illustration**). All these queries are diverse and have different characteristics. Query 1 does not involve any concepts from LOD cloud datasets and the mentioned terms are variables or concepts from upper level ontology. Query 2 is taken from Jamendo website. In the corresponding SPARQL query, apart from URI for "Punk" taken from Jamendo, the remaining terms are again either variables or concepts from upper level ontology. Query 3 involves processing results of queries on LOD datasets (USCensus and SemWebCorpus), which do not share a direct link in the LOD cloud. Thus, LOQUS can unify answers even when sub-query answers are not directly connected to each other. Query 4 (adopted from DARQ) is identical in spirit to Query 2 as it mentions specific LOD cloud concept (From DBpedia). However, the query utilizes information from a single source. This illustrates, that LOQUS can execute and process results for queries involving just one dataset as well.

Our results demonstrate that we are able to provide a mechanism to execute challenging queries on the LOD cloud without any compromise on execution time and by covering relevant datasets. The LOQUS approach also allows queries to retrieve and merge results which involve resources not directly connected to each other in the LOD cloud. Our evaluation shows that the LOQUS approach allows effective federation of SPARQL queries over the LOD cloud by using SUMO, a common upper level ontology. Using this approach we are able to answer queries, which cannot be answered by other state of the art systems for LOD query processing. Table 8.5 presents the various parameters on which LOQUS was evaluated for the three

no.	query	# results	Datasets	execution time(seconds)
Q1	Identify movies, countries where they were shot and the latest population for these countries.	1023	LinkedMDB, Geonames, DBpedia	80
Q2	Identify artists, whose albums have been tagged as punk and the population of the places they are based near.	54	Jamendo, MusicBrainz, Geonames, DBpedia	85
Q3	Identify congressional districts with active researchers in the area of Semantic Web.	30	SemWebCorpus, FOAF, Geonames, USCensus	110
Q4	Find name, birthday and image of German musicians born in Berlin.	8	DBpedia	65

Table 8.5: Result execution of queries using LOQUS

queries. Due to the restrictions imposed by SPARQL endpoints, the number of results returned for the query **may not** match the total number of entities available in datasets. The execution time has been averaged over 5 runs of the query.

Metric	LOQUS	DARQ	SQUIN
Requires user to know LOD Datasets	X	✓	✓
Approach	Uses upper level ontology (SUMO) for query serialization and execution.	Requires formal description of datasets in the form of Service Description.	Requires an initial URI to execute queries.
Query Creation	Creates query corresponding to every mapping for a concept.	Creates queries only corresponding to the concepts mentioned in the query.	Creates queries only corresponding to the concepts mentioned in the query.
Failsafe	Executes all subqueries for multiple mappings. Hence retrieves at least partial answers if a specific endpoint doesn't work.	X	X
Result Processing	Query answers, retrieved from different datasets are merged and presented to user.	Retrieves answers from only one dataset.	Retrieves answers from only one dataset.
Queries Answered	Q1,Q2,Q3,Q4	Q4	Q2,Q4

Table 8.6: Comparison LOD SPARQL Query Processing Systems

8.4.0.0.8 Qualitative comparison with other tools Table 8.6 compares LOQUS with DARQ and SQUIN on various parameters. The queries were executed for LOQUS. For other systems it is based on understanding of the capabilities of the system. DARQ [96] is a query engine which provides transparent query access to multiple, distributed SPARQL endpoints as if querying a single RDF graph which relies on "Service Descriptions" to specify the capabilities of a SPARQL endpoint. One of the limitations of DARQ is the use of predicates to decide the SPARQL endpoint to send triple patterns. Hence, it requires predicates to be bound. Thus it requires use of multiple queries to fetch results for Query 1 and Query 2. Absence of direct link between SemWebCorpus and USCensus, makes it impossible to fetch results for Query 3 using DARQ. SQUIN [53] allows LOD query answering by asynchronous traversal of RDF links to discover data that might be relevant for a query during the query execution itself. Hence, it requires at least one ground concept in the "subject" or "predicate" position of the triples contained in the query. Due to this requirement for crawling data, it is not possible to answer Query 1. Similarly Query 3 requires crawling to be performed from two different ends and then merging the crawled results and hence cannot be answered by SQUIN.

8.5 Related Work

To the best of our knowledge this is the first work presenting a system which allows users to query the LOD cloud without knowing the concepts from the diverse datasets and their interlinks. However, there are existing work on querying the LOD cloud which expects the user to know the concepts and the datasets which can answer the queries (introduced in the paper). These systems expect user to know the datasets and cannot answer the queries used for our evaluation. Another body of work which is related is the work in upper level ontology creation. A number of well known upper level ontologies such as SUMO [88], Cyc [99], and DOLCE [43] are available. In the past various domain specific ontologies have been integrated with these upper level ontologies [89, 34] driven by application specific needs. Other bodies of work relevant for this research is in the area of federation of database queries and schema matching and mapping.

8.6 Conclusion and Future Work

We have presented an approach for querying the LOD cloud without intimate knowledge of the individual datasets and the interconnecting relationships. Our results demonstrate that we are able to provide a mechanism to execute challenging queries on the LOD cloud without any compromise on execution time and by covering relevant datasets. The LOQUS approach allows automatic retrieval and merging of results for queries involving resources indirectly linked in the LOD cloud. Our evaluation shows LOQUS approach allows effective federation of SPARQL queries over the LOD cloud by using SUMO, a common upper level ontology. Using this approach we are able to answer queries, which cannot be answered by state of the art systems for LOD query processing.

Our future work includes extending the upper level ontology for including other datasets, analysis of query logs for better support of query answering and optimization of query plans for faster query execution. We also plan to release the querying system and upper level ontology as an open source project. We could not provide an online querying system as demonstrator for this submission due to the required anonymity. This will be added in the final version of the paper.

Conclusion

9.1 Summary

In this dissertation, we have presented the research issues and challenges related to data integration and querying in *big data* along with their solutions. These solutions have been implemented and evaluated using Linked Open Data (LOD) as benchmarks. We have also explained the importance of these issues in regards to fulfilling the vision of semantic web.

There are number of reasons why these issues arise which are covered in great detail in the chapters. Some of these issues are as follows:

- Most of the data being contributed on the LOD is coming from independent data publishers. There is little or no communication involved between these independent publishers and has resulted in heterogenous vocabularies being utilized for the same nature of data. In the absence of any explicit or implicit agreement, this process is bound to lead to heterogeneity in data and issues related to data integration.
- Another issue with LOD datasets is the massive reliance on converting relational data to RDF and eventually linked data. It is a well established fact that relational databases capture little or no semantics related to the data they store. Semantic web community makes an argument about differentiating itself from relational database community and capturing semantics of the data. Given that, it is rather ironic that tools and techniques like D2RQ and R2RML are the primary contributor of the data on the LOD cloud. Needless to say there is a lack of proper expression of semantics in this data. Hence,

the Linked Data community is facing the same issues related to data integration, which database community has been facing for a while.

- A related issue is the lack of established framework for querying of these independent and federated datasets. A good chunk of the solutions for querying of federated relational datasets rely on using an overarching schema for expressing queries. The lack of proper schema knowledge along with lack of overarching schema makes the querying problem more challenging and difficult to handle.

To alleviate these and other issues the dissertation has provided the following contributions:

- The dissertation provides a conceptual framework for using crowd generated data to identify semantic relationships between entities. Most of the previous works in the field of ontology matching depend on using linguistics based or rule based techniques for the purpose of relationship identification. The techniques presented in this dissertation are unique as they rely on noisy data generated independently by people to alleviate the issues plaguing the LOD cloud. These techniques have been implemented by BLOOMS [61], BLOOMS+ [67] and PLATO systems.
- The PLATO approach [62] is the only system of its kind which allows for identification of part-of relationships between entities. Further it also identifies the six different kind of part-of relationships presented in [118]. This approach utilizes data from both within the structured web as well as un-structured web to identify this relationship.
- Using the relationships identified by the approaches in the previous two items, a schema can be generated for the purpose of reasoning and proper knowledge representation. This schema can be utilized for the purpose of query answering and processing without knowing the individual datasets. This approach makes it easier to query and identify relevant knowledge from the LOD datasets. The applicability and ease of this approach has been demonstrated using the LOQUS system [64].
- The applicability of these approaches has been validated empirically using real data from Linked Data cloud. The experiments demonstrate that these approaches are scalable and work well even in case of noisy data.

The work though is a small step in the overall idea of using information contained within the datasets to improve the data. The approach has a potential to be used for solving numerous other research challenges. The subsequent section summarizes some of the challenges which can be solved using the idea of bootstrapping based approach.

9.2 Further Work

This dissertation has allowed for new methodologies for data integration and querying. These solutions have been demonstrated to be promising and can help in alleviating some of the issues plaguing the big and linked data cloud community. These solutions will open new areas for further research some of which are described as follows

9.2.1 Richer Relationship Identification on LOD

Using external knowledge and the existing relationships such as owl:sameAs, it is possible to identify richer and broader sets of relationships. This has two important implications for query answering that implicitly requires integration of LOD datasets: (i) it elicits more refined mapping between properties, and (ii) it enables aggregating and ranking triples from different LOD datasets.

For concreteness, consider the following example: There is an owl:sameAs link between the entity representing the movie *The Shining* on both LinkedMDB ¹ and DBpedia. Although they are talking about the same entity, the two databases provide different information about the entity. While LinkedMDB says *Jack Nicholson* is an actor in the film, DBpedia says *Jack Nicholson* is starring in the movie. By processing the equality of entity of interest (*The Shining*) and the value *Jack Nicholson*, it becomes possible to hypothesize the possible logical connection between the properties *actor* and *starring* between the two different schemas. This capability enhances the ability to answer broader set of questions. Thus, instead of limiting the vocabulary of users to a controlled and potentially unnatural subset from user's perspective, we can allow flexible phrasing of questions due to the availability of mapping between the relations.

¹<http://linkedmdb.org>

The work presented in this dissertation is a step in the direction of richer relationship identification. Identification of *owl:sameAs* and *part-of* relationship is just scratching the surface of this field and more work is necessary towards this field.

9.2.2 Yellow Pages for LOD

The number of datasets in LOD has grown above 300. However, there is a lack of a conceptual description of these datasets which makes it difficult to identify the datasets which can provide knowledge related to a specific domain. For example, a person looking for information related to *parasites* has no systematic way of identifying these datasets. In the absence of such systems it is difficult to perform query execution, information gathering and search on lod. While systems such as sig.ma [112] allow for search on lod, they do it only by finding resource(s) which contain a particular phrase.

There is a need for finding and ranking dataset based on their relevance to a specific term and/or domain. For this, a bootstrapping based approach can help greatly as a dataset with diverse domain and rich taxonomy such as Freebase ², DBpedia [18] can be used for acting as a conceptual descriptor of the datasets. Using terms from these datasets, different datasets can be described and these terms can be used as pointers for the datasets. Using these pointers applications can traverse and consume the appropriate datasets depending on the application. We have performed limited testing of this approach and the initial results on around 40 different LOD datasets are rather promising. Using this approach, we have been able to automatically identify that BBC Music is related to the domain of music, entertainment, artists. Similar kind of evaluation has been done on other datasets such as Diseasesome, NASA, Ordnance Survey Dataset. The ultimate objective of this work is to provide a search engine for finding appropriate datasets for a given domain and provide an ontology similar to voID [1] but at a more conceptual level.

9.2.3 Flexible Question Answering using LOD

Question and Answering (Q&A systems) have been around for a while, their use has been limited due to various requirements related to tight adherence to a specific format. To eliminate the issue of tight adherence

²<http://www.freebase.com/>

to the format of Q&A system, an expressive query language and its implementation can be provided to alleviate this issue. This further requires: (a) Development of semantic query graph (SQG) notation that is natural and easy to use conceptually, and convenient to translate to SPARQL automatically. (b) Mapping of concepts, properties, and triples in the user query to those available on the LOD, and (c) Selection of the appropriate LOD datasets and flexible ranking of answer sets.

The mapping of concepts and properties will be realized by extending the capabilities of our existing concept alignment systems, BLOOMS and BLOOMS+ [67, 61, 60] to generate better quality translations of query concepts and properties. The mapping of triples to the appropriate LOD datasets will exploit contextual information such as co-occurrence statistics, background knowledge, and provenance information. The SQG input will be modified through relevance feedback mechanism.

9.2.4 Property Matching on LOD

Ontology alignment systems have predominantly focussed on variants of isA-relationship among concepts (e.g., `rdfs:subclassOf`, `owl:equivalentClass`, and `owl:sameAs`). It is also important to learn isA-relationship among properties (e.g., `rdfs:subPropertyOf`) and other types of relationships among concepts (such as `partonymy`[65], `causes` [50, 103] etc) for developing natural and expressive queries.

To motivate this requirement, consider an example that shows how different property names are used to express the same relationship in two datasets in LOD. Consider the statement "Professor X advises doctoral student Y". In Dbpedia, this is modeled using a property name called `doctoralStudent` where professor X (`dbpedia:Willis_Lamb`) is connected to student Y (`dbpedia:Theodore_Maiman`) with a property called `doctoralStudent`. In Freebase the same relationship is modeled using a completely different property called `education.academic.advisees`. In Freebase the same professor X (`fbase:Willis_Lamb`) is connected to the very same student Y (`fbase:theodore.harold.maiman`) using `education.academic.advisees` as the connecting property. Although the two properties are semantically equivalent, it is difficult to see the similarity of them using simple techniques such as string matching over property names. Therefore a solution that goes beyond matching similar property names is required. There is a need for an algorithm that can identify similar property names which cannot be matched using already existing techniques for property mapping.

The algorithm can utilize owl:sameAs relationships between two datasets to match property names in them. In doing so, the occurrence of similar subject and object values across datasets is analyzed. Given any two matching subject and object pairs which are connected with owl:sameAs links, the binding property names of those subjects-objects pairs may exhibit equivalent relationships. The equivalence relationships between two property names can be examined by taking into account the frequency of matching subject-object pairs found for those two properties in a sample set. It is understandable that some values for subject and object may match randomly. But it can be imagined that the valid matches outnumber the random matches when the analysis is performed for large number of instances.

9.2.5 LOD Integration and Enhancement

The LOD cloud as an integration of various heterogeneous datasets presents users with the unique opportunity to study relationships between various entities belonging to diverse domains. However, the absence of an overarching ontology requires users to have an intimate knowledge of the cloud such as which sections of the cloud should be queried for answers to parts of a given query. To query the cloud is also challenging because it requires users to execute queries at different independent endpoints and then manually combine the answers. This issue can be resolved by enhancing the LOD cloud with schema knowledge.

An upper level ontology can be created that will fill the void for the requirement of an overarching integration of the LOD cloud. The ontology will consist of a schema and appropriate mappings to the schema of LOD datasets. This will provide users a single reference point to identify the relevant datasets, rather than expecting the user to inspect multiple independent overlapping datasets before formulating the query. For example, the upper level ontology will contain the knowledge that bills are legal documents and plurality vote as represented in GovTrack is a subclass of bill as the following representation illustrates.

$$\text{PluralityVote} \sqsubseteq \text{Bill} \quad (9.1)$$

This integration will significantly improve information integration, data cleaning and entity resolution. For example, it will allow a user to query for `Bill` in the upper level ontology (or even for `Law`, which would be a superclass of `Bill`), and the query system can then automatically specialize this to a query

of `PluralityVote` for `GovTrack`, thus releasing the user from the burden of knowing about specific representations within LOD datasets.

It is also important to realize that this upper level ontology does not need to be created from scratch, since there are already good existing ontologies which can be used for this purpose. SUMO [88], due to its broad thematic coverage, seems to be an excellent starting point.

BLOOMS can be a centerpiece in the creation of the upper level ontology from existing schema knowledge and ontologies. BLOOMS can be enhanced for all aspects related to upper ontology creation and query federation. In particular, support can be added for (1) other thematic domains which are insufficiently covered by Wikipedia, (2) mapping of properties and creation of property hierarchies, and (3) alignment of instance data. These enhancements follow naturally from the breakthrough results reported in [61], by complementing Wikipedia, as used in BLOOMS, by other resources such as WordNet or Cyc, and by systematically continuing the exploitation of the use of such resources for ontology alignment (which is the novel aspect in BLOOMS, on which its high performance rests).

Thus, BLOOMS will enable releasing an upper level ontology for LOD querying, which will essentially consist of a class hierarchy and a property hierarchy, for query federation.

9.3 Final Remarks

In conclusion, the dissertation has presented approaches for data integration, relationship identification and querying in scenarios where data is disparate, massive and created independently by various publishers. The dissertation has demonstrated that community generated data though noisy is still powerful and useful for identification of semantic relationship such as subsumption and partonomy between entities.

Secondly, these relationships once established can be utilized for effectively answering queries over this data. This querying approach relies on the schema created by the relationship identification techniques identified above. Hence, there is a value in creating better models for knowledge representation rather than just creating richer instance base.

The approaches presented in dissertation work have also been validated on Linked Open Data Cloud

datasets. Our experiments demonstrate that semantic relationships can be identified between these datasets using the approaches identified above. Further, the querying approach allows querying of these datasets without knowing the individual datasets and schemas.

Bibliography

- [1] ALEXANDER, K., CYGANIAK, R., HAUSENBLAS, M., AND ZHAO, J. Describing Linked Datasets – On the Design and Usage of void, the 'Vocabulary of Interlinked Datasets'. In *WWW2009 Workshop on Linked Data on the Web (LDOW2009)* (Madrid, Spain, 2009).
- [2] ALKHATEEB, F., BAGET, J.-F., AND EUZENAT, J. Extending sparql with regular expression patterns (for querying rdf). *Web Semant.* 7, 2 (Apr. 2009), 57–73.
- [3] ANYANWU, K., MADUKO, A., AND SHETH, A. Sparq2l: towards support for subgraph extraction queries in rdf databases. In *Proceedings of the 16th international conference on World Wide Web* (New York, NY, USA, 2007), WWW '07, ACM, pp. 797–806.
- [4] ARISTOTLE. *The Organon*. Shenandoah Bible Ministries, 2009.
- [5] ARTALE, A., FRANCONI, E., GUARINO, N., AND PAZZI, L. Part-whole relations in object-centered systems: An overview. *Data & Knowledge Engineering* 20, 3 (1996), 347–383.
- [6] BAKER, C., FILLMORE, C., AND LOWE, J. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1* (1998), Association for Computational Linguistics, Morgan Kaufmann Publishers, pp. 86–90.
- [7] BATINI, C., LENZERINI, M., AND NAVATHE, S. B. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.* 18, 4 (Dec. 1986), 323–364.
- [8] BECKETT, D. RDF/XML Syntax Specification (Revised), February 2004.
- [9] BECKETT, D., AND BERNERS-LEE, T. Turtle - terse RDF triple language, W3C team submission, 2008. See: <http://www.w3.org/TeamSubmission/turtle/>.
- [10] BERGMAN, M. K., AND GIASSON, F. Umbel ontology, volume 1, technical documentation. Tech. rep., Structured Dynamics LLC, 2008. Available from http://www.umbel.org/doc/UMBELOntology_vA1.pdf.
- [11] BERGMAN, M. K., AND GIASSON, F. UMBEL ontology, volume 1, technical documentation. Tech. Rep. 1, Structured Dynamics, 2008. Available from: http://umbel.org/doc/UMBELOntology_vA1.pdf.
- [12] BERNERS-LEE, T., FIELDING, R. T., AND MASINTER, L. Uniform resource identifier (URI): Generic syntax. RFC 3986, RFC Editor, Fremont, CA, USA, Jan. 2005.

- [13] BERNERS-LEE, T., AND FISCHETTI, M. *Weaving the web: The original design and ultimate destiny of the world wide web by its inventor*. Harper, San Francisco, 1999.
- [14] BERNERS-LEE, T., AND HENDLER, J. Scientific publishing on the semantic web. *Nature* 410 (2001), 1023–1024.
- [15] BERNERS-LEE, T., HENDLER, J., LASSILA, O., ET AL. The Semantic Web. *Scientific American* 284, 5 (2001), 28–37.
- [16] BIZER, C., CYGANIAK, R., AND HEATH, T. How to publish linked data on the web. Web page, 2007. Revised 2008. Accessed 22/02/2010.
- [17] BIZER, C., HEATH, T., AND BERNERS LEE, T. Linked data - the story so far. *International Journal on Semantic Web and Information Systems* 5, 3 (2009), 1–22.
- [18] BIZER, C., LEHMANN, J., KOBILAROV, G., AUER, S., BECKER, C., CYGANIAK, R., AND HELLMANN, S. DBpedia—A crystallization point for the Web of Data. *Journal of Web Semantics* 7, 3 (2009), 154–165.
- [19] BOUQUET, P., STOERMER, H., NIEDEREE, C., AND MA NA, A. Entity name system: The backbone of an open and scalable web of data. In *ICSC '08: Proceedings of the 2008 IEEE International Conference on Semantic Computing* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 554–561.
- [20] CARLSON, A., BETTERIDGE, J., KISIEL, B., SETTLES, B., JR., E. R. H., AND MITCHELL, T. M. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)* (2010).
- [21] CASATI, R., AND VARZI, A. *Parts and places: The structures of spatial representation*. The MIT Press, 1999.
- [22] CHAUDHRY, O., AND MACKANESS, W. Utilising partonomic information in the creation of hierarchical geographies. In *10th ICA Workshop on Generalisation and Multiple Representation. Moscow, Russia* (2007).
- [23] CHOI, N., SONG, I.-Y., AND HAN, H. A survey on ontology mapping. *SIGMOD Rec.* 35, 3 (2006), 34–41.
- [24] CIMIANO, P., HOTH, A., AND STAAB, S. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Int. Res.* 24 (August 2005), 305–339.
- [25] CODD, E. F. A relational model of data for large shared data banks. *Commun. ACM* 26, 1 (Jan. 1983), 64–69.
- [26] COLLINS, A. M., AND QUILLIAN, M. R. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior* 8, 2 (1969), 240 – 247.
- [27] COLLINS, A. M., AND QUILLIAN, M. R. Does category size affect categorization time? *Journal of Verbal Learning and Verbal Behavior* 9, 4 (1970), 432 – 438.
- [28] CORNET, R., AND DE KEIZER, N. Forty years of SNOMED: a literature review. *BMC medical informatics and decision making* 8 Suppl 1 (2008).

- [29] CRUZ, I., MENDELZON, A., AND WOOD, P. G+: Recursive Queries without Recursion. In *Proceedings of the 2nd Expert Database Systems Conference* (1989), Benjamin-Cummings, pp. 645–666.
- [30] CRUZ, I. F., MENDELZON, A. O., AND WOOD, P. T. A graphical query language supporting recursion. In *Proceedings of the 1987 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 1987), SIGMOD '87, ACM, pp. 323–330.
- [31] DAMOVA, M., KIRYAKOV, A., SIMOV, K., AND PETROV, S. Mapping the Central LOD Ontologies to PROTON Upper-Level Ontology. In *In Proceedings of the Fifth International Workshop on Ontology Matching* (November 2010), P. Shvaiko, J. Euzenat, F. Giunchiglia, H. Stuckenschmidt, M. Mao, and I. Cruz, Eds., CEUR Workshop Proceedings.
- [32] DAVID, J., EUZENAT, J., SCHARFFE, F., AND DOS SANTOS, C. T. The alignment api 4.0. *Semantic Web 2*, 1 (2011), 3–10.
- [33] DAVID, J., GUILLET, F., AND BRIAND, H. Matching directories and OWL ontologies with AROMA. In *CIKM '06: Proceedings of the 15th ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2006), ACM, pp. 830–831.
- [34] DE MELO, G., SUCHANEK, F., AND PEASE, A. Integrating YAGO into the Suggested Upper Merged Ontology. In *ICTAI 2008* (2008).
- [35] EGENHOFER, M. J. Toward the semantic geospatial web. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems* (New York, NY, USA, 2002), GIS '02, ACM, pp. 1–4.
- [36] EUZENAT, J. An API for ontology alignment. In *Proceedings of Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004* (2004), S. A. McIlraith, D. Plexousakis, and F. van Harmelen, Eds., vol. 3298 of *Lecture Notes in Computer Science*, Springer, pp. 698–712.
- [37] EUZENAT, J., D'AQUIN, M., SABOU, M., AND ZIMMER, A. Matching ontologies for context. Tech. Rep. NEON/2006/D3.3.1/0.8, INRIA, 2007.
- [38] EUZENAT, J., AND SHVAIKO, P. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
- [39] FELLBAUM, C., Ed. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*, illustrated edition ed. The MIT Press, May 1998.
- [40] FENSEL, D., HENDLER, J. A., LIEBERMAN, H., WAHLSTER, W., AND BERNERS-LEE, T. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. The MIT Press, 2005.
- [41] FERRUCCI, D., BROWN, E., CHU-CARROLL, J., FAN, J., GONDEK, D., KALYANPUR, A. A., LALLY, A., MURDOCK, J. W., NYBERG, E., AND PRAGER, J. Building watson: An overview of the deepqa project. *AI Magazine* 31, 3 (2010), 59–79.
- [42] FERRUCCI, D. A. Introduction to : This is watson. *IBM Journal of Research and Development* 56, 3.4 (may-june 2012), 1:1 –1:15.
- [43] GANGEMI, A., GUARINO, N., MASOLO, C., OLTRAMARI, A., AND SCHNEIDER, L. Sweetening Ontologies with DOLCE. In *EKAW 2002* (2002), vol. 2473 of *LNCS*, pp. 223–233.
- [44] GERSTL, P., AND PRIBBENOW, S. A conceptual theory of part-whole relations and its applications. *Data & Knowledge Engineering* 20, 3 (1996), 305–322.

- [45] GIRJU, R., BADULESCU, A., AND MOLDOVAN, D. Automatic discovery of part-whole relations. *Computational Linguistics* 32, 1 (2006), 83–135.
- [46] GIUNCHIGLIA, F., AUTAYEU, A., AND PANE, J. S-Match: an open source framework for matching lightweight ontologies, 2010.
- [47] GIUNCHIGLIA, F., SHVAIKO, P., AND YATSKEVICH, M. S-Match: an algorithm and an implementation of semantic matching. In *Semantic Interoperability and Integration* (Dagstuhl, Germany, 2005), Y. Kalfoglou, M. Schorlemmer, A. Sheth, S. Staab, and M. Uschold, Eds., no. 04391 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.
- [48] GLASER, H. Museum data, where next ? consuming linked data. Event Dates: 2011-01-13, January 2011.
- [49] GRUBER, T., ET AL. Toward principles for the design of ontologies used for knowledge sharing. *International journal of human computer studies* 43, 5 (1995), 907–928.
- [50] GU, P. Causal knowledge modeling for traditional chinese medicine using owl 2. In *9th International Semantic Web Conference (ISWC2010)* (November 2010).
- [51] HALEVY, A. Y. Answering queries using views: A survey. *The VLDB Journal* 10, 4 (Dec. 2001), 270–294.
- [52] HAMDI, F., SAFAR, B., NIRLAULA, N. B., AND REYNAUD, C. Taxomap in the OAEI 2009 Alignment Contest. In *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) at the 8th International Semantic Web Conference (ISWC-2009) Chantilly, USA, October 25, 2009* (2009), P. Shvaiko, J. Euzenat, F. Giunchiglia, H. Stuckenschmidt, N. F. Noy, and A. Rosenthal, Eds. Available from: http://ceur-ws.org/Vol-551/oeai09_paper14.pdf.
- [53] HARTIG, O., BIZER, C., AND FREYTAG, J.-C. Executing SPARQL Queries over the Web of Linked Data. In *Proceedings of the 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009* (2009), A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, Eds., vol. 5823 of *LNCS*, Springer, pp. 293–309.
- [54] HARTIG, O., AND HEESE, R. The sparql query graph model for query optimization. In *Proceedings of the 4th European conference on The Semantic Web: Research and Applications* (Berlin, Heidelberg, 2007), ESWC '07, Springer-Verlag, pp. 564–578.
- [55] HAUSENBLAS, M. Exploiting linked data to build web applications. *IEEE Internet Computing* 13 (2009), 68–73.
- [56] HEARST, M. A. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics – Volume 2* (Stroudsburg, PA, USA, 1992), COLING '92, pp. 539–545.
- [57] HITZLER, P. Towards Reasoning Pragmatics. In *Proceedings of the Third International Conference on GeoSpatial Semantics* (2009), K. Janowicz, M. Raubal, and S. Levashkin, Eds., Springer.
- [58] HITZLER, P., KRÖTZSCH, M., PARSIA, B., PATEL-SCHNEIDER, P., AND RUDOLPH, S., Eds. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at <http://www.w3.org/TR/owl2-primer/>.

- [59] HORROCKS, I., PATEL-SCHNEIDER, P. F., BOLEY, H., TABET, S., GROSOFF, B., AND DEAN, M. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C Member Submission 21 May 2004, 2004. Available from <http://www.w3.org/Submission/SWRL/>.
- [60] JAIN, P., HITZLER, P., AND SHETH, A. P. Flexible bootstrapping-based ontology alignment. In *Proceedings of the 5th International Workshop on Ontology Matching (OM-2010)* (2010), P. Shvaiko, J. Euzenat, F. Giunchiglia, H. Stuckenschmidt, M. Mao, and I. Cruz, Eds., CEUR-WS.
- [61] JAIN, P., HITZLER, P., SHETH, A. P., VERMA, K., AND YEH, P. Z. Ontology Alignment for Linked Open Data. In *Proceedings of the 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010* (2010), P. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Pan, I. Horrocks, and B. Glimm, Eds., vol. 6496, Springer-Verlag, pp. 402–417. Available from <http://data.semanticweb.org/conference/iswc/2010/paper/218>.
- [62] JAIN, P., HITZLER, P., VERMA, K., YEH, P. Z., AND SHETH, A. P. Moving beyond sameas with plato: partonomy detection for linked data. In *Proceedings of the 23rd ACM conference on Hypertext and social media* (New York, NY, USA, June 2012), E. V. Munson and M. Strohmaier, Eds., HT '12, ACM, pp. 33–42.
- [63] JAIN, P., HITZLER, P., YEH, P. Z., VERMA, K., AND SHETH, A. P. Linked Data is Merely More Data. In *Linked Data Meets Artificial Intelligence* (2010), AAAI Press, Menlo Park, CA, pp. 82–86.
- [64] JAIN, P., VERMA, K., YEH, P. Z., HITZLER, P., AND SHETH, A. P. LOQUS: Linked Open Data SPARQL Querying System. Tech. rep., Kno.e.sis Center, Wright State University, Dayton, Ohio, 2010. Available from <http://www.pascal-hitzler.de/resources/publications/loqus-tr-2010.pdf>.
- [65] JAIN, P., YEH, P., VERMA, K., HENSON, C., AND SHETH, A. SPARQL Query Re-writing for Spatial Datasets Using Partonomy Based Transformation Rules. In *Proceedings of the Third International Conference on GeoSpatial Semantics* (2009), Springer, pp. 140–158.
- [66] JAIN, P., YEH, P. Z., VERMA, K., HENSON, C. A., AND SHETH, A. P. SPARQL query re-writing using partonomy based transformation rules. In *Proceedings of the 3rd International Conference on GeoSpatial Semantics* (Berlin, Heidelberg, 2009), GeoS '09, Springer-Verlag, pp. 140–158.
- [67] JAIN, P., YEH, P. Z., VERMA, K., VASQUEZ, R. G., DAMOVA, M., HITZLER, P., AND SHETH, A. P. Contextual ontology alignment of lod with an upper ontology: a case study with proton. In *Proceedings of the 8th extended semantic web conference on The semantic web: research and applications - Volume Part I* (Berlin, Heidelberg, 2011), ESWC'11, Springer-Verlag, pp. 80–92.
- [68] JEAN-MARY, Y. R., SHIRONOSHITA, E. P., AND KABUKA, M. R. Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web* 7, 3 (2009), 235 – 251. `¡ce:title¿The Web of Data¡ce:title¿`.
- [69] JING, Y., JEONG, D., AND BAIK, D. Sparql graph pattern rewriting for owl-dl inference query. In *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on* (2008), vol. 2, IEEE, pp. 675–680.
- [70] KOLAS, D. Supporting spatial semantics with sparql. *Transactions in GIS* 12 (2008), 5–18.
- [71] KOLAS, D., AND SELF, T. Spatially-augmented knowledgebase. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference* (Berlin, Heidelberg, 2007), ISWC'07/ASWC'07, Springer-Verlag, pp. 792–801.

- [72] KRISNADHI, A., MAIER, F., AND HITZLER, P. OWL and Rules. In *Reasoning Web. Semantic Technologies for the Web of Data – 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures* (2011), vol. 6848 of *Lecture Notes in Computer Science*, Springer, Heidelberg, pp. 382–415.
- [73] KRÖTZSCH, M., MAIER, F., KRISNADHI, A. A., AND HITZLER, P. A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In *Proceedings of the 20th International World Wide Web Conference, WWW2011, Hyderabad, India, March/April 2011* (2011), ACM, New York, pp. 645–654.
- [74] LEHMANN, J., AND HITZLER, P. Concept learning in description logics using refinement operators. *Machine Learning* 78, 1–2 (2010), 203–250.
- [75] LI, J., TANG, J., LI, Y., AND LUO, Q. RiMOM: A dynamic multistrategy ontology alignment framework. *IEEE Transactions on Knowledge and Data Engineering* 21 (2009), 1218–1232.
- [76] LITWIN, W., AND ABDELLATIF, A. Multidatabase interoperability. *Computer* 19, 12 (Dec. 1986), 10–18.
- [77] MADHAVAN, J., BERNSTEIN, P. A., AND RAHM, E. Generic schema matching with Cupid. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 2001), Morgan Kaufmann Publishers Inc., pp. 49–58.
- [78] MANYIKA, J., CHUI, M., BROWN, B., BUGHIN, J., DOBBS, R., ROXBURGH, C., AND BYERS, A. H. Big data: The next frontier for innovation, competition, and productivity, May 2011.
- [79] MÀRQUEZ, L., AND RODRÍGUEZ, H. Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the 10th European Conference on Machine Learning (ECML-98)* (Berlin, Apr. 1998), C. Nédellec and C. Rouveirol, Eds., vol. 1398 of *LNAI*, Springer, pp. 25–36.
- [80] MASCARDI, V., LOCORO, A., AND ROSSO, P. Automatic Ontology Matching via Upper Ontologies: A Systematic Evaluation. *IEEE Transactions on Knowledge and Data Engineering* 22, 5 (2010), 609–623.
- [81] MASOLO, C., BORGO, S., GANGEMI, A., GUARINO, N., OLTRAMARI, A., AND SCHNEIDER, L. The WonderWeb library of foundational ontologies. *LADSEB-Cnr, Padova, IT, Preliminary Report D 17* (2002). Available from <http://wonderweb.semanticweb.org/deliverables/D17.shtml>.
- [82] MEDELYAN, O., AND LEGG, C. Integrating Cyc and Wikipedia: Folksonomy meets rigorously defined common-sense. In *Proceedings of the AAAI Workshop on Wikipedia and Artificial Intelligence, Chicago, Illinois, USA* (2008), R. Bunescu, E. Gabrilovich, and R. Mihalcea, Eds., pp. 13–18.
- [83] MILLER, E., AND MANOLA, F. RDF primer. W3C recommendation, W3C, Feb. 2004. <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>.
- [84] MOTIK, B., PATEL-SCHNEIDER, P., AND PARSIA, B., Eds. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. W3C Recommendation, 27 October 2009. Available at <http://www.w3.org/TR/owl2-syntax/>.
- [85] MOTIK, B., SATTTLER, U., AND STUDER, R. Query answering for OWL DL with rules. *Journal of Web Semantics* 3, 1 (2005), 41–60.

- [86] MURDOCK, J. W., KALYANPUR, A., WELTY, C., FAN, J., FERRUCCI, D. A., GONDEK, D. C., ZHANG, L., AND KANAYAMA, H. Typing candidate answers using type coercion. *IBM Journal of Research and Development* 56, 3.4 (may-june 2012), 7:1–7:13.
- [87] NIKOLOV, A., UREN, V. S., MOTTA, E., AND ROECK, A. N. D. Overcoming schema heterogeneity between linked semantic repositories to improve coreference resolution. In *The Semantic Web, Fourth Asian Conference, ASWC 2009, Shanghai, China, December 6-9, 2009. Proceedings* (2009), vol. 5926, Springer, pp. 332–346.
- [88] NILES, I., AND PEASE, A. Towards a Standard Upper Ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems – Volume 2001* (2001), pp. 2–9.
- [89] OBERLE, D., ET AL. DOLCE ergo SUMO: On Foundational and Domain Models in the SmartWeb Integrated Ontology (SWIntO). *JWS* 5, 3 (2007), 156–174.
- [90] PANTEL, P., AND PENNACCHIOTTI, M. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics* (Stroudsburg, PA, USA, 2006), ACL-44, pp. 113–120.
- [91] PARUNDEKAR, R., KNOBLOCK, C., AND AMBITE, J. L. Linking and building ontologies of linked data. In *9th International Semantic Web Conference (ISWC2010)* (November 2010), P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks, and B. Glimm, Eds., vol. 6496 of *Lecture Notes in Computer Science*, Springer, pp. 589–614.
- [92] PERRY, M., SHETH, A. P., HAKIMPOUR, F., AND JAIN, P. Supporting complex thematic, spatial and temporal queries over semantic web data. In *Proceedings of the 2nd international conference on GeoSpatial semantics* (Berlin, Heidelberg, 2007), GeoS’07, Springer-Verlag, pp. 228–246.
- [93] PONZETTO, S. P., AND NAVIGLI, R. Large-scale taxonomy mapping for restructuring and integrating wikipedia. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009* (2009), C. Boutilier, Ed., pp. 2083–2088.
- [94] PONZETTO, S. P., AND STRUBE, M. Deriving a large scale taxonomy from Wikipedia. In *AAAI’07: Proceedings of the 22nd National Conference on Artificial Intelligence* (2007), AAAI Press, pp. 1440–1445.
- [95] P©REZ, J., ARENAS, M., AND GUTIERREZ, C. nsparql: A navigational language for rdf. In *The Semantic Web - ISWC 2008*, A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, and K. Thirunarayan, Eds., vol. 5318 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 66–81.
- [96] QUILITZ, B., AND LESER, U. Querying Distributed RDF Data Sources with SPARQL. In *Proceedings of the 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008*, (2008), S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, Eds., vol. 5021 of *LNCS*, Springer, pp. 524–538.
- [97] RAHM, E., AND BERNSTEIN, P. A. A survey of approaches to automatic schema matching. *The VLDB Journal* 10, 4 (2001), 334–350.
- [98] RECTOR, A., WELTY, C., NOY, N., AND WALLACE, E. Simple part-whole relations in OWL Ontologies available at <http://www.w3.org/2001/sw/bestpractices/oep/simplepartwhole/>, Aug. 2005.

- [99] REED, S., AND LENAT, D. Mapping Ontologies into Cyc. Tech. rep., Cycorp, Inc., 2002. Available from http://www.cyc.com/doc/white_papers/.
- [100] SEABORNE, A., AND PRUDHOMMEAU, E. SPARQL query language for RDF, 2008. <http://www.w3.org/TR/rdf-sparql-query/>.
- [101] SHETH, A., ARPINAR, I. B., AND KASHYAP, V. Relationships at the heart of semantic web: Modeling, discovering, and exploiting complex semantic relationships. In *Enhancing the Power of the Internet Studies in Fuzziness and Soft Computing* (2003), Springer-Verlag, pp. 63–94.
- [102] SHETH, A., LARSON, J., CORNELIO, A., AND NAVATHE, S. A tool for integrating conceptual schemas and user views. In *Data Engineering, 1988. Proceedings. Fourth International Conference on* (feb 1988), pp. 176–183.
- [103] SHETH, A., THACKER, S., AND PATEL, S. Complex relationships and knowledge discovery support in the infoquilt system. *The VLDB Journal* 12 (May 2003), 2–27.
- [104] SIRIN, E., AND PARSIA, B. Sparql-dl: Sparql query for owl-dl. In *In 3rd OWL Experiences and Directions Workshop (OWLED-2007)* (2007).
- [105] SMITH, B. The basic tools of formal ontology. In *Formal Ontology in Information Systems* (1998).
- [106] SMITH, B., CEUSTERS, W., KLAGGES, B., KÖHLER, J., KUMAR, A., LOMAX, J., MUNGALL, C., NEUHAUS, F., RECTOR, A., AND ROSSE, C. Relations in biomedical ontologies. *Genome Biology* 6 (2005), R46.
- [107] SPILIOPOULOS, V., VALARAKOS, A. G., AND VOUIROS, G. A. Csr: Discovering subsumption relations for the alignment of ontologies. In *Proceedings of 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008* (2008), S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, Eds., vol. 5021 of *Lecture Notes in Computer Science*, Springer, pp. 418–431.
- [108] SUCHANEK, F. M., KASNECI, G., AND WEIKUM, G. Yago: A Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007* (New York, NY, USA, 2007), C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, Eds., ACM Press.
- [109] TERZIEV, I., KIRYAKOV, A., AND MANOV, D. Base upper-level ontology (bulo) Guidance. Tech. rep., Ontotext Lab, Sirma Group, 2004. Deliverable of EU-IST Project IST-2003-506826, Available from http://proton.semanticweb.org/D1_8_1.pdf.
- [110] TOUPIKOV, N., UMBRICH, J., DELBRU, R., HAUSENBLAS, M., AND TUMMARELLO, G. DING! Dataset ranking using formal descriptions. In *WWW2009 Workshop on Linked Data on the Web (LDOW2009)* (2009).
- [111] TRYFONA, N., AND EGENHOFER, M. J. Consistency among parts and aggregates: A computational model. *Transactions in GIS* 1, 3 (1996), 189–206.
- [112] TUMMARELLO, G., CYGANIAK, R., CATASTA, M., DANIELCZYK, S., DELBRU, R., AND DECKER, S. Sig.ma: Live views on the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web* 8, 4 (2010), 355 – 364. [jce:title;Semantic Web Challenge 2009j/ce:title; jce:title;User Interaction in Semantic Web researchj/ce:title;](#)

- [113] VAN HAGE, W., KOLB, H., AND SCHREIBER, G. A method for learning part-whole relations. In *The Semantic Web - ISWC 2006*, vol. 4273 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, pp. 723–735.
- [114] VARZI, A. Parts, wholes, and part-whole relations: The prospects of mereotopology. *Data & Knowledge Engineering* 20, 3 (1996), 259–286.
- [115] VOEGELE, T., HUEBNER, S., AND SCHUSTER, G. Buster - an information broker for the semantic web. *KI* 17, 3 (2003), 31.
- [116] VOLZ, J., BIZER, C., GAEDKE, M., AND KOBILAROV, G. Discovering and maintaining links on the web of data. In *ISWC '09: Proceedings of the 8th International Semantic Web Conference* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 650–665.
- [117] VOLZ, J., BIZER, C., GAEDKE, M., AND KOBILAROV, G. Silk—A Link Discovery Framework for the Web of Data. In *2nd Linked Data on the Web Workshop (LDOW2009)* (Madrid, Spain, 2009). Available from http://ceur-ws.org/Vol-538/ldow2009_paper13.pdf.
- [118] WINSTON, M. E., CHAFFIN, R., AND HERRMANN, D. A taxonomy of part-whole relations. *Cognitive Science* 11, 4 (1987), 417–444.