Exploiting Alignments in Linked Data for Compression and Query Answering

Amit Krishna Joshi

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
https://corescholar.libraries.wright.edu/etd_all/1766

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.
Exploiting Alignments in Linked Data for Compression and Query Answering

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

by

AMIT KRISHNA JOSHI
B.E., Institute of Engineering, Nepal, 2004
M.Sc., University of Reading, UK, 2008

2017
Wright State University
WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

March 28, 2017

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Amit Krishna Joshi ENTITLED Exploiting Alignments in Linked Data for Compression and Query Answering BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

______________________________
Pascal Hitzler, Ph.D.
Dissertation Director

______________________________
Michael L. Raymer, Ph.D.
Director, Computer Science and Engineering Ph.D. Program

______________________________
Robert E.W. Fyffe, Ph.D.
Vice President for Research and Dean of the Graduate School

Committee on
Final Examination

______________________________
Pascal Hitzler, Ph.D.

______________________________
Guozhu Dong, Ph.D.

______________________________
Krishnaprasad Thirunaraya, Ph.D.

______________________________
Michelle Cheatham, Ph.D.

______________________________
Subhashini Ganapathy, Ph.D.
Linked data has experienced accelerated growth in recent years due to its interlinking ability across disparate sources, made possible via machine-processable RDF data. Today, a large number of organizations, including governments and news providers, publish data in RDF format, inviting developers to build useful applications through reuse and integration of structured data. This has led to tremendous increase in the amount of RDF data on the web. Although the growth of RDF data can be viewed as a positive sign for semantic web initiatives, it causes performance bottlenecks for RDF data management systems that store and provide access to data. In addition, a growing number of ontologies and vocabularies make retrieving data a challenging task.

The aim of this research is to show how alignments in the Linked Data can be exploited to compress and query the linked datasets. First, we introduce two compression techniques that compress RDF datasets through identification and removal of semantic and contextual redundancies in linked data. Logical Linked Data Compression is a lossless compression technique which compresses a dataset by generating a set of new logical rules from the dataset and removing triples that can be inferred from these rules. Contextual Linked Data Compression is a lossy compression technique which compresses datasets by performing schema alignment and instance matching followed by pruning of alignments based on confidence value and subsequent grouping of equivalent terms. Depending on the structure of the dataset, the first technique was able to prune more than 50% of the triples. Second, we propose an Alignment based Linked Open Data Querying System (ALOQUS) that allows users to write query statements using concepts and properties not present in linked datasets and show that querying does not require a thorough understanding of the individual datasets and interconnecting relationships. Finally, we present LinkGen, a multipurpose synthetic Linked Data generator that generates a large amount of repeatable and reproducible RDF data using statistical distribution, and interlinks with real world entities using alignments.
Contents

1 Introduction 1
   1.1 RDF Compression 2
   1.2 Query Answering 3
   1.3 Ontology Alignment 4
   1.4 Research Statements 5
   1.5 Contributions 5
   1.6 Dissertation Overview 6

2 Background 7
   2.1 Semantic Web and Linked Data 7
   2.2 Data Interchange: RDF 9
      2.2.1 RDF/XML 11
      2.2.2 N-Triples 12
      2.2.3 Turtle 12
      2.2.4 N-Quads 13
      2.2.5 TriG 13
      2.2.6 JSON-LD 14
   2.3 Query Language: SPARQL 14
   2.4 Ontology 16
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5 Alignment API Format</td>
<td>16</td>
</tr>
<tr>
<td>2.6 Upper Ontology</td>
<td>18</td>
</tr>
<tr>
<td>2.7 Dictionary Encoding</td>
<td>18</td>
</tr>
<tr>
<td>2.8 Data Compression</td>
<td>20</td>
</tr>
<tr>
<td>2.9 Development Tools and Technologies</td>
<td>21</td>
</tr>
<tr>
<td>3 Logical Linked Data Compression</td>
<td>22</td>
</tr>
<tr>
<td>3.1 Frequent Itemset Mining</td>
<td>23</td>
</tr>
<tr>
<td>3.1.1 Association Rule Mining</td>
<td>26</td>
</tr>
<tr>
<td>3.1.2 Multi-Dimensional Association Rules</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Rule Based Compression</td>
<td>28</td>
</tr>
<tr>
<td>3.2.1 Intra-property RB Compression</td>
<td>29</td>
</tr>
<tr>
<td>3.2.2 Inter-property RB Compression</td>
<td>31</td>
</tr>
<tr>
<td>3.2.3 Optimal Frequent Patterns</td>
<td>32</td>
</tr>
<tr>
<td>3.2.4 Delta Compression</td>
<td>33</td>
</tr>
<tr>
<td>3.3 Decompression</td>
<td>34</td>
</tr>
<tr>
<td>3.4 Evaluation</td>
<td>34</td>
</tr>
<tr>
<td>3.4.1 RB Compression - Triple Reduction</td>
<td>34</td>
</tr>
<tr>
<td>3.4.2 Comparison using compressed dataset size</td>
<td>36</td>
</tr>
<tr>
<td>3.4.3 RB Compression on Benchmark Dataset</td>
<td>37</td>
</tr>
<tr>
<td>3.5 Soundness and Completeness</td>
<td>37</td>
</tr>
<tr>
<td>4 Contextual Linked Data Compression</td>
<td>40</td>
</tr>
<tr>
<td>4.1 Ontology Alignments</td>
<td>40</td>
</tr>
<tr>
<td>4.1.1 Schema Alignment</td>
<td>41</td>
</tr>
<tr>
<td>4.1.2 Instance Matching</td>
<td>41</td>
</tr>
<tr>
<td>4.2 Approach</td>
<td>42</td>
</tr>
<tr>
<td>4.3 Evaluation</td>
<td>44</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>4.3.1</td>
<td>OAEI Conference Ontology</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Dataset Generation</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Varied Alignments and Compression</td>
</tr>
<tr>
<td>4.4</td>
<td>Discussion</td>
</tr>
<tr>
<td>5</td>
<td>Alignment based Linked Open Data Querying System</td>
</tr>
<tr>
<td>5.1</td>
<td>Linked Open Data and Data Retrieval</td>
</tr>
<tr>
<td>5.2</td>
<td>Challenges</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Intimate knowledge of datasets</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Schema heterogeneity</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Entity Co-reference</td>
</tr>
<tr>
<td>5.3</td>
<td>Approach</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Automatic mapping between upper level Ontology and Ontologies used in LOD Datasets</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Identification and mapping of concepts in user defined queries to those in LOD Datasets</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Constructing Sub-queries</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Execution of sub-Queries</td>
</tr>
<tr>
<td>5.3.5</td>
<td>Determining entity co-references</td>
</tr>
<tr>
<td>5.3.6</td>
<td>Transformation and local storage of RDF graphs</td>
</tr>
<tr>
<td>5.3.7</td>
<td>Joining and Processing of results</td>
</tr>
<tr>
<td>5.4</td>
<td>Scenario Illustration</td>
</tr>
<tr>
<td>5.5</td>
<td>Evaluation</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Statement and Query Types</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Queries and Results</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Qualitative comparison with other tools</td>
</tr>
<tr>
<td>5.6</td>
<td>Discussion</td>
</tr>
</tbody>
</table>
6 Synthetic Linked Data Generator

6.1 Synthetic Linked Data .............................................. 71
6.2 Data Generator Features .......................................... 73
  6.2.1 Entity Distribution ............................................. 73
  6.2.2 Noisy Data ..................................................... 73
  6.2.3 Inter-linking real world entities ............................. 74
  6.2.4 Output Data and Streaming mode ............................. 75
  6.2.5 Config Parameters ............................................. 75
  6.2.6 Data Generation Steps ........................................ 75
6.3 Evaluation .......................................................... 77
6.4 Conclusion .......................................................... 78

7 Related Work

7.1 RDF Compression .................................................. 79
  7.1.1 Adjacency List .................................................. 80
  7.1.2 Bitmap .......................................................... 81
  7.1.3 HDT ............................................................. 82
7.2 Querying LOD ....................................................... 83
7.3 Synthetic Linked Data Generator ................................. 84

8 Conclusion

8.1 Alignments and RDF compression ............................... 86
8.2 Alignments and Query Answering ............................... 87
8.3 Alignments and Synthetic Data Generation ..................... 88
8.4 Future Work ........................................................ 88

Bibliography
# List of Figures

2.1 Latest Semantic Web Layer Stack ........................................... 8
2.3 Example of RDF Graph data .................................................. 10
2.4 SPARQL query to fetch five oldest US president ......................... 15
2.5 SPARQL query response in Turtle format for query listed in 2.4. ........ 15
2.6 Ontology snippet showing T-Box and A-Box .............................. 17
2.7 Alignment API format example showing ‘map’ element ................. 17
2.8 Hierarchy example in SUMO .................................................. 19
2.9 Triples encoded using numeric IDs ......................................... 19
3.1 Rule Based Compression ...................................................... 23
3.2 List of encoded triples and corresponding transactions .................. 24
3.3 Rule Based Compression, $G = G_D \cup R(G_A)$ .......................... 28
3.4 Compression and Decompression time for various linked open datasets . 35
3.5 Compression and Decompression time for various LUBM datasets ........ 38
4.1 Conceptual System Overview ................................................. 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Grouping equivalent terms for ekaw#Regular_Paper and ekaw#Research_Topic using OAEI reference alignment.</td>
<td>43</td>
</tr>
<tr>
<td>4.3</td>
<td>OAEI Conference Track Ontologies</td>
<td>45</td>
</tr>
<tr>
<td>4.4</td>
<td>Dataset size for various set of queries.</td>
<td>46</td>
</tr>
<tr>
<td>4.5</td>
<td>Varying alignment for same pair of items.</td>
<td>47</td>
</tr>
<tr>
<td>4.6</td>
<td>Number of mappings at different thresholds for two versions of Conference reference alignments</td>
<td>48</td>
</tr>
<tr>
<td>4.7</td>
<td>Comparison of various automated alignment systems demonstrating varying number of equivalent terms for same threshold</td>
<td>48</td>
</tr>
<tr>
<td>4.8</td>
<td>Compressed size (in MB) against original size of 670MB</td>
<td>49</td>
</tr>
<tr>
<td>5.1</td>
<td>LinkedMdb connects to DBpedia via NYTimes</td>
<td>53</td>
</tr>
<tr>
<td>5.2</td>
<td>ALOQUS Illustration</td>
<td>59</td>
</tr>
<tr>
<td>6.1</td>
<td>Power-law distribution of subjects in Wikipedia</td>
<td>74</td>
</tr>
<tr>
<td>6.2</td>
<td>Time taken for generating datasets of various sizes</td>
<td>78</td>
</tr>
<tr>
<td>7.1</td>
<td>List of Triples</td>
<td>80</td>
</tr>
<tr>
<td>7.2</td>
<td>Dictionary encoding for terms in 7.1</td>
<td>80</td>
</tr>
<tr>
<td>7.3</td>
<td>Compact Transformation from ID-based triples using adjacency list</td>
<td>81</td>
</tr>
<tr>
<td>7.4</td>
<td>Bitmap Transformation from Compact Streams</td>
<td>82</td>
</tr>
<tr>
<td>7.5</td>
<td>Dictionary and three possible triple representations for triples in 7.1</td>
<td>82</td>
</tr>
</tbody>
</table>
# List of Tables

2.1 Dictionary encoding for terms appearing in triples listed in Section 2.2 . . . 20

3.1 Numerically encoded item and corresponding term . . . . . . . . . . . . . 26

3.2 Sample frequent patterns generated for *DBpedia Ontology Types* dataset.
   An item can be associated with multiple frequent patterns as seen for item
   70. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26

3.3 Frequent patterns generated for the Geonames dataset. Each item is a pair
   of property and object \( (p : o) \). . . . . . . . . . . . . . . . . . . . . . . . . . 27

3.4 Compression ratio for various linked open datasets . . . . . . . . . . . . . 35

3.5 Comparison of various compression techniques based on dataset size . . . . 37

3.6 Compression ratio and time for various LUBM datasets . . . . . . . . . . . 38

5.1 Result containing proxy identifiers . . . . . . . . . . . . . . . . . . . . . . . 62

5.2 Result containing LOD identifiers . . . . . . . . . . . . . . . . . . . . . . . 63

5.3 ALOQUUS Queries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66

5.4 Comparison of LOD SPARQL Query Processing Systems . . . . . . . . . . . 68

6.1 Properties with no domain or range info in DBpedia ontology . . . . . . . . 76

6.2 Characteristics of the datasets used for evaluation . . . . . . . . . . . . . 77
# List of Algorithms

1. Intra-property RB compression .......................... 30
2. Inter-property RB compression .......................... 31
3. Delta Compression ....................................... 33
4. Consolidation of Alignments .......................... 44
Acknowledgments

I would like to express my sincere gratitude towards my advisor Prof. Pascal Hitzler for his support, guidance, and encouragement without which this thesis would not have been possible. I would like to express my special appreciation to members of my dissertation committee: Dr. Guozhu Dong, Dr. Krishnaprasad Thirunaraya, Dr. Michelle Cheatham and Dr. Subhashini Ganapathy for their cooperation and encouragement. A special mention goes to Dr. Guozhu Dong for the invaluable discussions and suggestions to improve this dissertation work.

I would also like to take this opportunity to thank my professors and friends in the Department of Computer Science and Engineering for their help and support during my graduate studies and their genuine interest in the success of my work. Special thanks to my friends in the Data Semantics Lab, Cloud Computing Lab and Kno.e.sis Lab for their feedback, suggestions and ideas. I would also like to thank the administrative staff in our department for their help and cooperation throughout my time at Wright State.

My family has been a constant source of support for me. I thank my wife Kripa for her understanding and love. I am thankful to my son Aadit for always making me smile. I thank my parents, brothers and sisters for their support and encouragement in my endeavors.

I would also like to acknowledge the support of various funding sources. Wright State University for providing the tuition waiver and travel support. The National Science Foundation for support through NSF project award 1017225 “II: Small: TROn – Tractable Reasoning with Ontologies.”, NSF project award 1143717 "III: EAGER Expressive Scalable Querying over Integrated Linked Open Data." and NSF travel awards for participating at various conferences.
1 Introduction

The term “Linked Data”, coined by Tim Berners-Lee in his Design Issues: Linked Data note [BL11], refers to a set of rules for publishing and interlinking structured data on the web. The set of rules, now known as principles of Linked Data, is built upon the idea of web of data similar to the current web of documents and employs web standards to the task of sharing data on a global scale. The four rules of linked data are:

1. Use URIs as identifiers for things.

2. Use HTTP URIs so that these identifiers/things can be looked up

3. Provide useful information about the identifier when it’s looked up, using standards (ex: RDF, SPARQL)

4. Include links to other related URIs so that they can discover more things.

The adoption of the Linked Data principles has led to the extension of web with a massive growth in structured data, shared and interlinked across diverse domains such as people, companies, books, films, music, medicine, e-commerce and online communities [BHBL09]. Prominent data publishers such as The New York Times,\(^1\) the US government,\(^2\) the UK government,\(^3\) BBC Music,\(^4\) and PubMed\(^5\) have adopted this methodology.

\(^1\)http://data.nytimes.com/home/
\(^2\)http://data.gov
\(^3\)http://data.gov.uk/data
\(^4\)http://www.bbc.co.uk/music
\(^5\)http://www.ncbi.nlm.nih.gov/pubmed/
to interlink their data. Over the last decade, Web of data has grown from less than a dozen of datasets to more than 9000 datasets\(^6\) with more than 140 billions of facts expressed in RDF (Resource Description Framework) format. Similarly, Linked Open Vocabularies (LOV)\(^7\) dataset now consists of more than 500 vocabularies, 20,000 classes and almost 30,000 properties.

Although the growth of RDF data can be viewed as a positive sign for semantic web initiatives, it causes performance bottlenecks for RDF data management systems that store and provide access to data. As such, the need for compressing structured data is becoming increasingly important. In addition, a growing number of ontologies and vocabularies make retrieving data a challenging task.

### 1.1 RDF Compression

The key to RDF compression is to understand the syntax and semantics of the RDF data model [KC04] and identify redundancies to represent the same information in a compact manner. The RDF data model represents information as simple directed graphs, designed to facilitate the exchange of information among diverse applications and multiple sources at internet scale. In RDF, a description of a resource is represented as a collection of triples, each consisting of a subject, a predicate and an object. A triple can be represented as a node-arc-node link with arc pointing towards the object.

Real world RDF statistics\(^8\) reveal a continuous increase in RDF data as well as great diversity in the domain and source of information. [FGMP10a] studied the RDF compressibility capabilities and observed that big real world RDF datasets are highly compressible due to the skewed structure of the graphs, organization of URIs and syntax verbosity. The skewed characteristics of real world datasets are discussed in the studies of [DF06],

---

\(^6\)http://lod-cloud.net/state/

\(^7\)http://lov.okfn.org/dataset/lov

\(^8\)http://stats.lod2.eu/
Various RDF representations and compression techniques have been developed to reduce the size of RDF data for storage and transmission. Representation like N3 [BLC98], Turtle [BBLP08] and RDF/JSON [Ale08] offer compactness while maintaining readability by reducing the verbosity of the original RDF/XML format [BM04]. Earlier RDF compression studies [FGMP10a, ACZH10, UMD+13, FMPG+13] focused on dictionary encoding and compact representation of RDF triples. [FMPG+13] proposed a binary serialization format called Header-Dictionary-Triples (HDT) that takes advantage of redundancies and skewed data in large RDF graphs. [IPR05] introduced the notion of a lean graph, which is obtained by eliminating triples with blank nodes that specify redundant information. [PPSW10] and [Mei08] studied the problem of redundancy elimination on RDF graphs in the presence of rules and constraints.

1.2 Query Answering

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 with data about music, movies, reviews, scientific publications, government information, geographical locations, medicine and many more. To take advantage of the enormously extensive structured data in the LOD cloud, one must be able to effectively pose queries to and retrieve answers from it. However, a growing number of ontologies and vocabularies make retrieving data a challenging task.

Querying the LOD cloud requires users to have a thorough understanding of various concepts and datasets prior to creating a query. For example, consider the query “Identify films, the nations where they were shot and the populations of these countries.” Answering this query requires a 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 practical given the size (and continued growth) of the LOD cloud. Furthermore, issues such as schema heterogeneity and entity disambiguation identified in [JHY+10] present profound challenges with respect to querying of the LOD cloud. Each of these data sources can be queried separately, most often through an end point using the SPARQL query language [PS+06]. Looking for answers making use of information spanning over different data sets is a more challenging task as the mechanisms used internally to query datasets (database-like joins, query planning) cannot be easily generalized to this setting.

1.3 Ontology Alignment

Ontology Alignment [NS05] refers to the task of finding correspondences between ontologies. It’s a widely explored topic and numerous applications have been developed that perform the task of ontology alignment and mapping for schemas and instances. OAEI⁹ regularly organizes the evaluation of various ontology matching systems [ES+07, EMS+11b] and compare them based on formal tests and datasets. Ontology alignment plays an important role whenever one attempts to integrate data from multiple sources. It finds a number of use cases including schema integration, data integration, ontology evolution, agent communication and query answering on the web [Noy04, JJH+12]. Alignments are not limited to one-to-one equivalent relationships. They can be of various cardinalities: 1:1 (one-to-one), 1:m (one-to-many), n:1 (many-to-one) or n:m (many-to-many) and of various relationships e.g., equivalence, disjointness or subsumption [SE13]. Since LOD caters to multiple domains, Ontology Alignment plays a major role in integrating data from multiple datasets within LOD cloud. This study aims to exploit the alignments in linked data to perform compression and query answering.

⁹http://oaei.ontologymatching.org/
1.4 Research Statements

Research statements for our work are as follows:

1. Lossless compression of RDF datasets can be achieved by mining logical rules present in the RDF dataset.

2. Contextual lossy compression can be achieved by exploiting schema alignment and instance matching among linked datasets.

3. Queries can be written in upper level ontologies instead of domain specific ontologies for querying against linked datasets.

1.5 Contributions

The principle contribution of this study are listed below.

- Algorithm for automatic generation of rules for achieving lossless compression is developed, implemented and evaluated.

- Algorithm for delta compression, referring to an incremental change in dataset, is presented.

- Strategy to determine optical frequent pattern to achieve greater compression is provided.

- Lossy compression of RDF datasets has been introduced and evaluated using OAEI Conference ontology and reference alignments.

- Challenges in querying LOD have been identified and listed.

- Alignment based querying approach is introduced and necessary steps to perform such querying are provided along with scenario illustration.
• To distinguish queries, various Statement and Query types are defined.

• LinkGen, a platform independent synthetic linked data generator, is developed

1.6 Dissertation Overview

The structure of this dissertation is as follows. In Chapter 2, we provide a background on Linked Data and various key topics related to our research. In Chapter 3, we dive into details of our Rule based compression (RBC) with comprehensive evaluation performed against state-of-the art compression systems. In Chapter 4, we provide a conceptual overview of our proposed alignment aware compression and highlight the role of alignments and contexts in RDF compression. In Chapter 5, we discuss the approach for alignment based querying system and provide end-to-end usage scenario. In Chapter 6, we discuss the importance of synthetic data generator and present our multipurpose RDF data generator tool, LinkGen. In Chapter 7, we provide details on prior works related to our research work. Finally, we conclude with summary of the dissertation study and future work ideas in Chapter 8.
2 Background

2.1 Semantic Web and Linked Data

In today’s information society, the World Wide Web plays a key role for disseminating and retrieving information. Most of the information available on the web is published in the form of hypertext documents and has been designed primarily for human consumption. As such, data used for creating such documents either remain in traditional databases or in textual form within documents making it difficult for sharing and reuse. Semantic Web [BLHL+01] is an extension of current web which focuses on data exchange and interoperability. Figure 2.1 depicts the latest architecture of semantic web, which contains a set of open standards arranged in the form of layers referred as Semantic Layer Cake. The goal of semantic web is to move beyond current "Web of Documents" and create a "Web of Data" that can be processed by machines to perform more useful tasks, such as intelligently searching and combining data from heterogeneous dataset. To achieve this, it is important to have huge amount of structured data represented using standard format. In addition, these data need to be accessible as well as interlinked using open standards. A collection of such connected and inter-related datasets can be referred to as Linked Data. Principles of Linked Data are described in Section 1.

In recent years, there is an increased momentum in publishing datasets using Linked

---

1 Image copied from https://www.w3.org/2007/03/layerCake.png
Data principles. Furthermore, there’s a rise in number of open datasets which are published using linked data principles. Open data refers to data which is freely available and can be consumed by anyone. Such a collection of open datasets published using Linked Data principles is referred to as Linked Open Data (LOD). The recent statistics\(^2\) reveals that there are more than 9000 datasets in LOD cloud publishing more than 140 billion triples. Figure 2.2 shows connections among various Linked Open Datasets differentiated by dataset categories such as Geography and Government.

One of the most widely used linked open dataset is DBpedia, a community effort to extract structured information from Wikipedia and to make this information available on the Web [ABK+07].

\(^2\)http://stats.lod2.eu/
2.2 Data Interchange: RDF

Resource Description Framework (RDF) [HM04] is a W3C standard for describing resources on the Web. It allows authors to describe any resource that can be uniquely identified by means of a URI\(^3\). The base element of an RDF model is an RDF triple, and a set of RDF triples is known as an RDF graph.

---

\(^3\)http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#section-Graph-URIref
Definition 1. Let $U$, $B$, $L$ represent the disjoint sets of RDF URI references, blank nodes and literals respectively. We call a triple of the form $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ an RDF triple, where $s$ represents the subject, $p$ the predicate and $o$ the object.

In an RDF graph, a set of subjects and objects represent a set of nodes while a set of predicates represents relationships between nodes. An RDF graph is a directed graph where relationship always points towards the object. The predicate is also known as the property of the triple. Figure 2.3 shows a sample RDF graph where $s1$, $s2$ and $s3$ denotes subjects and all the connections are for one single property 'rdf:type'.

Definition 2. Two RDF graphs $G$ and $G'$ are equivalent if there is a bijection $M$ between the sets of triple terms of the two graphs, such that:

1. $M$ maps blank nodes to blank nodes.
2. $M(lit) = lit$ for all RDF literals $lit$ which are either nodes or edges of $G$.
3. $M(uri) = uri$ for all RDF URI references $uri$ which are either nodes or edges of $G$.
4. The triple $(s, p, o)$ is in $G$ if and only if the triple $(M(s), M(p), M(o))$ is in $G'$.
A number of serialization formats exists for representing RDF graphs. However, using different serialization formats lead to exactly the same triples, and they are thus logically equivalent. The following sub-sections provide details on most widely used formats for serializing RDF data. RDF graphs can also be embedded in web pages using various formats such as RDFa\(^4\) and microdata\(^5\).

### 2.2.1 RDF/XML

RDF/XML [BM04] is the first standard format for serializing RDF using a widely popular XML notation. In RDF/XML, RDF triples are specified within an XML element rdf:RDF. The XML element rdf:Description\(^6\) is used to define sets of triples for a subject specified by rdf:about attribute. The following XML document includes few triples about Abraham Lincoln.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dbp="http://dbpedia.org/property/"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdf:Description rdf:about="http://dbpedia.org/resource/Abraham_Lincoln">
    <rdf:type rdf:resource="http://dbpedia.org/ontology/Person" />
    <rdfs:label xml:lang="en">Abraham Lincoln</rdfs:label>
    <dbp:birthPlace xml:lang="en">Hodgenville, Kentucky, U.S.</dbp:birthPlace>
    <dbp:birthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1809-02-12</dbp:birthDate>
    <dbp:deathDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1865-04-15</dbp:deathDate>
    <dbp:spouse rdf:resource="http://dbpedia.org/resource/Mary_Todd_Lincoln" />
  </rdf:Description>
</rdf:RDF>
```

\(^4\)http://www.w3.org/TR/rdfa-lite/

\(^5\)http://schema.org/docs/gs.html#microdata_how

\(^6\)http://www.w3.org/1999/02/22-rdf-syntax-ns#Description
2.2.2 N-Triples

N-Triples [BB01] provides a simple line-based text serialization of RDF graphs. Each line contains one triple with RDF terms separated by white space. Each triple is terminated by a dot (‘.’). RDF blank nodes in N-Triples are expressed as `_:` followed by a blank node label.

```
```

2.2.3 Turtle

Turtle [BBLP08] is a compact textual representation of an RDF graph. It is an extension of N-Triples and supports namespace prefixes, lists and shorthands for datatyped literals.

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbp: <http://dbpedia.org/property/> .

dbr:Abraham_Lincoln rdf:type dbo:Person ;
  rdfs:label "Abraham Lincoln"@en ;
  dbo:birthPlace "Hodgenville, Kentucky, U.S."@en ;
  dbo:birthDate "1809-02-12"^^xsd:date ;
  dbo:children "Robert, Edward, Willie, and Tad"@en ;
  dbo:deathDate "1865-04-15"^^xsd:date ;
  dbo:spouse dbr:Mary_Todd_Lincoln .
```


### 2.2.4 N-Quads

N-Quads [Car14] is a simple extension to N-Triples that allows one to add a fourth element to a line to provide context for a triple. This optional context element can be either a blank node label or a graph IRI of the triple described on that line.

```
_:subject1 <http://an.example/predicate1> "object1" <http://example.org/graph1> .
_:subject2 <http://an.example/predicate2> "object2" <http://example.org/graph2> .
```

### 2.2.5 TriG

Turtle syntax supports only the specification of single graphs. TriG [CS14] is an extension of Turtle and supports the specification of multiple graphs. Simple Triples, Predicate Lists, and Object Lists can all be used either inside a graph statement, or on their own as in a Turtle document. Triples written outside a graph statement will be a part of an unnamed graph i.e. 'default' graph.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbp: <http://dbpedia.org/property/> .

# default graph - no {} used.
<http://example.org/al> rdfs:label "Abraham Lincoln"@en ;

# GRAPH keyword. Abbreviation of triples using ;
GRAPH <http://example.org/al/>
{
[
] rdf:type dbo:Person ;
  rdfs:label "Abraham Lincoln"@en ;
  dbp:birthPlace "Hodgenville, Kentucky, U.S."@en ;
  dbp:birthDate "1809-02-12"^^xsd:date ;
}
```
2.2.6 JSON-LD

JSON-LD [SLK+14] serializes RDF graphs and datasets in JSON\footnote{http://www.rfc-editor.org/info/rfc7159} format. It’s part of W3C recommendation specification and is primarily targeted for Web applications. Since it’s compatible with JSON, a large number of existing parsers can be reused.

```
{
    "@context": "example-context.json",
    "@id": "http://example.org/bob#me",
    "@type": "Person",
    "birthdate": "1990-07-04",
    "knows": "http://example.org/alice#me",
    "interest": {
        "@id": "http://www.wikidata.org/entity/Q12418",
        "title": "Mona Lisa",
        "subject_of": "http://data.europeana.eu/item/04802",
        "creator": "http://dbpedia.org/resource/Leonardo_da_Vinci"
    }
}
```

2.3 Query Language: SPARQL

SPARQL defines a standard query language and data access protocol for querying and manipulating RDF data [PS+06]. It uses graph pattern matching technique to return the matched triple patterns. In addition to the basic graph matching, it also supports optional pattern matching which provides the flexibility to include the additional patterns if present (without eliminating the solution). SPARQL can query across federated datasets and supports various features including join, filter, sub-queries and sorting, similar to SQL for relational databases. Four primary query forms of SPARQL are SELECT, ASK, DESCRIBE and CONSTRUCT. SPARQL supports graph update using various update operations in-
Figure 2.4: SPARQL query to fetch five oldest US president

```sparql
PREFIX purl: <http://purl.org/dc/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX dbp: <http://dbpedia.org/property/>
SELECT ?president, ?birthday WHERE {
  ?president purl:subject dbc:Presidents_of_the_United_States.
  ?president dbp:birthDate ?birthday.
} ORDER BY ?birthday
LIMIT 5
```

Figure 2.5: SPARQL query response in Turtle format for query listed in 2.4.

including INSERT, DELETE, LOAD, CLEAR and COPY. It also supports service description\(^8\) which provides a mechanism to discover information about SPARQL services such as available dataset and supported extension functions. A SPARQL endpoint is an URI used for querying a RDF dataset via the SPARQL language. The results of SPARQL queries can be obtained in various formats including JSON, RDF, XML and HTML.

Figure 2.4 shows the result of 2.4 executed against the DBpedia SPARQL endpoint\(^9\).

\(^8\)https://www.w3.org/TR/2013/REC-sparql11-service-description-20130321/
\(^9\)https://dbpedia.org/sparql
2.4 Ontology

[Gru93] defined ontology as "an explicit specification of conceptualization". Ontologies are used to define the concepts and the relationships that exist in a domain of interest. In practice, ontologies can range from simpler ones with few terms to a very complex one consisting of thousands of terms such as SNOMEDCT\textsuperscript{10}. The term Vocabulary is often used instead of ontology to refer to a collection of identifiers with clearly defined meanings [HKR11]. RDFS [BG04] and OWL[HKP+09] are used for representing and exchanging ontologies. RDFS [BG04] is simply another RDF vocabulary that provides language constructs in the form of class and property hierarchies and their semantic interdependencies. RDF and RDFS vocabulary are suitable for modeling only simple ontologies as they have limited expressivity [HKR11]. In order to model complex knowledge, OWL (Web Ontology Language) which is more expressive is used.

Ontology contains information about the schema and instances. Schema, also known as T-Box, refers to classes, properties and any restrictions on them whereas instance data, also known as A-Box, refer to the assertions about the individuals made against the schema. Statements in T-Box are generally static whereas statements in A-Box keep evolving.

In the Figure 2.4, all statements except the last one about Product_Laptop\_xyz are T-Box statements.

2.5 Alignment API Format

In order to facilitate sharing of alignment across various applications, including those consuming alignments and generating alignments, a common format called Alignment API format [Euz04] was developed. Although a more expressive Alignment format, EDOAL,
has been recently developed, we limit our discussion to a general Alignment format, which is also the format used by OAEI for their reference alignments. The main element of this format is the ‘map’ element, shown in Figure 2.7, that provides the relation type between two entities in different ontologies and also provides a measure value denoting the confidence that the relation holds. It allows indicating various relationships namely equivalence, subsumes, isSubsumed, hasInstance and InstanceOf.

Figure 2.6: Ontology snippet showing T-Box and A-Box

Figure 2.7: Alignment API format example showing ‘map’ element
2.6 Upper Ontology

Upper Level Ontology refers to an ontology that provides definitions for general terms that are common across multiple domains. The primary objective of an upper level ontology is to establish a link among a large number of domain specific ontologies to increase semantic interoperability. SUMO (Suggested Upper Merged Ontology)$^{11}$ is one such upper level ontology that contains definitions for the most general concepts and the relations between them [NP01]. It has mappings to all of WordNet and is extended with many domain ontologies including ontologies of Food, Sports, Music, Hotels and Geography. Figure 2.6 depicts an entity hierarchy example$^{12}$ in SUMO for a geographic area. PROTON[TKM05] is another upper ontology which cover most of the upper-level concepts, about 300 classes and 100 properties, necessary for semantic annotation, indexing, and retrieval. Other upper level ontologies include Basic Formal Ontology (BFO)$^{13}$, Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE)$^{14}$, General Formal Ontology (GFO)$^{15}$ and Cyc upper ontology$^{16}$.

2.7 Dictionary Encoding

In general terms, a data dictionary is a centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format [McD94]. Since the RDF format is too verbose, the first step performed before manipulating RDF data is dictionary encoding. This component assigns a unique ID to each subject, predicate and object in the dataset. The process is straightforward and can be done sequentially or in parallel

---

$^{11}$http://www.adampease.org/OP/
$^{13}$http://www.ifomis.org/bfo/
$^{14}$http://www.loa.istc.cnr.it/old/DOLCE.html
$^{15}$http://www.onto-med.de/ontologies/gfo
$^{16}$http://www.cyc.com/kb/
[UMD⁺13] for fast processing. Once IDs are generated, RDF triples can be encoded by replacing each element in triples with its corresponding IDs from the dictionary. The list of triples represented in Turtle format in Section 2.2.3 can be encoded in several ways, one of which is listed in 2.9, based on the dictionary 2.7. A data structure such as an adjacency list can be subsequently used for compact representation of these triples.

1 101 3.
1 106 4.
1 102 5.
1 103 6.
1 104 4.
1 105 8.
1 107 2.

Figure 2.9: Triples encoded using numeric IDs
<table>
<thead>
<tr>
<th>ID</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>db:Abraham</td>
</tr>
<tr>
<td>2</td>
<td>db:Mary_Todd_Lincoln</td>
</tr>
<tr>
<td>3</td>
<td>dbo:Person1</td>
</tr>
<tr>
<td>4</td>
<td>&quot;Abraham Lincoln&quot;@en</td>
</tr>
<tr>
<td>5</td>
<td>&quot;Hodgenville, Kentucky, U.S.&quot;@en</td>
</tr>
<tr>
<td>6</td>
<td>&quot;1809-02-12&quot;^^xsd:date</td>
</tr>
<tr>
<td>7</td>
<td>&quot;Robert, Edward, Willie, and Tad&quot;@en</td>
</tr>
<tr>
<td>8</td>
<td>&quot;1865-04-15&quot;^^xsd:date</td>
</tr>
</tbody>
</table>

| 101 | rdf:type                             |
| 102 | dbp:birthPlace                       |
| 103 | dbp:birthDate                        |
| 104 | dbp:children                         |
| 105 | dbp:deathDate                        |
| 106 | rdfs:label                           |
| 107 | dbp:spouse                           |

Table 2.1: Dictionary encoding for terms appearing in triples listed in Section 2.2

### 2.8 Data Compression

Data compression refers to the process of minimizing the amount of data needed to store or transmit. It involves transforming a string of characters in some representation (such as ASCII) into a new string (of bits, for example) which contains the same information but whose length is smaller than the original [LH87]. Compression can be either lossy or lossless. Lossless compression reduces bits by identifying and eliminating redundancy in such a way that the losslessly compressed data can be decompressed to exactly its original value. Lossy compression involves removing 'unimportant' and non-essential data and as such represents an approximation of the original message. Lossless algorithms are typically used for text, and lossy for images, audio and video where a little bit of loss in information is often undetectable, or at least acceptable.
2.9 Development Tools and Technologies

All the applications developed for this work has been built using JAVA and various open source libraries. Tools and technologies used during development includes JENA\textsuperscript{17}, REDIS\textsuperscript{18}, Mahout\textsuperscript{19}, MapReduce and Hadoop\textsuperscript{20}, Alignment API Library\textsuperscript{21} and BLOOMS Alignment System\textsuperscript{22}.

\textsuperscript{17}http://jena.apache.org/
\textsuperscript{18}https://redis.io/
\textsuperscript{19}http://mahout.apache.org/
\textsuperscript{20}http://hadoop.apache.org/docs/r1.0.4/mapred_tutorial.html
\textsuperscript{21}http://alignapi.gforge.inria.fr/lib.html
\textsuperscript{22}http://wiki.knoesis.org/index.php/BLOOMS
3 Logical Linked Data Compression

In this chapter, we introduce a lossless compression of RDF datasets using automatic generation of decomposition rules. We have devised an algorithm to automatically generate a set of rules and split the database into two smaller disjoint datasets, viz., an Active dataset and a Dormant dataset based on those rules. The dormant dataset contains a list of triples which remain uncompressed and to which no rule can be applied during decompression. On the other hand, the active dataset contains list of compressed triples, to which rules are applied for inferring new triples during decompression. Figure 3.1 depicts the general system architecture of our system.

In order to automatically generate a set of rules for compression, we employ frequent pattern mining techniques [HPYM04, LWZ+08]. We examine two possibilities for frequent mining - a) within each property (hence, intra-property) and b) among multiple properties (inter-property). Experiments reveal that RB compression performs better when inter-property transactions are used instead of intra-property transactions. Specifically, the contribution of this work is a rule-based compression technique with the following properties:

- The compression reduces the number of triples, without introducing any new subjects, properties or objects.

- The set of decompression rules, $R$, can be automatically generated using widely used association rule mining techniques.
3.1 Frequent Itemset Mining

The concept of frequent itemset mining [AIS93] (FIM) was first introduced for mining transaction databases. Over the years, frequent itemset mining has played an important role in many data mining tasks that aim to find interesting patterns from databases, including association rules and correlations, or aim to use frequent itemsets to construct classifiers and clusters [Goe03]. In this study, we exploit frequent itemset mining techniques on RDF datasets for generating logical rules and subsequent compressing of RDF datasets.

Transaction Database.

Let \( I = \{i_1, i_2, \ldots, i_n\} \) be a set of distinct items. A set \( X = \{i_1, i_2, \ldots, i_k\} \subseteq I \) is called an itemset, or a k-itemset if it contains k items. Let \( D \) be a set of transactions where each transaction, \( T = (tid, X) \), contains a unique transaction identifier, \( tid \), and an itemset
X. Figure 3.1 shows a list of transactions corresponding to a list of triples containing the rdf:type\(^1\) property. Here, subjects represent identifiers and the set of corresponding objects represent transactions. In this study, we use the following definitions for intra- and inter-property transactions.

**Intra-property transactions.** For a graph \( G \) containing a set of triples, an intra-property transaction corresponding to a property \( p \) is a set \( T = (s, X) \) such that \( s \) is a subject and \( X \) is a set of objects, i.e. \((s, p, o_x)\) is a triple in graph \( G \); \( o_x \) is a member of \( X \).

**Inter-property transactions.** For a graph \( G \) containing a set of triples, an inter-property transaction is a set \( T = (s, Z) \) such that \( s \) is a subject and each member of \( Z \) is a pair \((p_z, o_z)\) of property and object, i.e. \((s, p_z, o_z)\) is a triple in graph \( G \).

<table>
<thead>
<tr>
<th>TID</th>
<th>rdf:type</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>125,22,225,60</td>
</tr>
<tr>
<td>S2</td>
<td>125,22,225</td>
</tr>
<tr>
<td>S3</td>
<td>81,22</td>
</tr>
<tr>
<td>S4</td>
<td>125,22,225,60</td>
</tr>
<tr>
<td>S5</td>
<td>125,22</td>
</tr>
<tr>
<td>S6</td>
<td>90,22</td>
</tr>
</tbody>
</table>

Figure 3.2: List of encoded triples and corresponding transactions

**Support and Frequent Itemset.**

The *support* of an itemset \( X \), denoted by \( \sigma(X) \), is the number of transactions in \( D \) containing \( X \). Itemset \( X \) is said to be *frequent* if \( \sigma(X) \geq \sigma_{\min} \) (\( \sigma_{\min} \) is a minimum support threshold).

\(^1\) rdf:type is represented by \( a \)
Itemset Mining.

A frequent itemset is often referred to as a *frequent pattern*. Numerous studies have been done and various algorithms [AIS93, AS94, HPYM04, SON95, SVA97] have been proposed to mine frequent itemsets. In this study, we use the *FP-Growth* [HPYM04] algorithm for generating frequent itemsets. We represent the output of FP-Growth as a set of pairs \((k, F_k)\), where \(k\) is an item, and \(F_k\), a set of frequent patterns corresponding to \(k\). Each frequent pattern is a pair of the form \((v, \sigma_v)\). \(v\) is an itemset of a frequent pattern and \(\sigma_v\) is a support of this frequent pattern.

**Definition 3.** Let \(D\) be a transaction database over a set \(I\) of items, and \(\sigma_{\text{min}}\) a minimum support threshold. The set of frequent itemsets in \(D\) with respect to \(\sigma_{\text{min}}\) is denoted by \(F(D, \sigma_{\text{min}}) := \{X \subseteq I | \sigma(X) \geq \sigma_{\text{min}}\}\)

Table 3.1 shows several frequent patterns for *DBpedia Ontology Types* dataset containing only the rdf:type property.\(^2\) To generate such frequent patterns, we first create a transaction database as shown in Figure 3.1 and then use parallel FP-Growth to compute frequent patterns. Please refer to [HPYM04, LWZ+08] for details about the FP-Growth algorithm and its implementation. Figure 3.3 shows the list of inter-property frequent patterns for Geonames dataset.

\(^2\)http://downloads.dbpedia.org/preview.php?file=3.7_sl_en_sl_instance_types_en.nt.bz2
Item | Object
---|---
22 | owl:Thing
227 | dbp:Work
189 | dbp:Film
213 | schema:Movie
103 | dbp:Person
26 | schema:Person
304 | foaf:Person
173 | dbp:Artist
225 | dbp:Place
60 | schema:Place

<table>
<thead>
<tr>
<th>Item ((k))</th>
<th>Frequent Patterns ((F_k))</th>
</tr>
</thead>
<tbody>
<tr>
<td>225</td>
<td>{([22, 225], 525786)}</td>
</tr>
<tr>
<td>60</td>
<td>{([22, 225, 60], 525786)}</td>
</tr>
<tr>
<td>189</td>
<td>{([22, 227, 83, 189], 60194)}</td>
</tr>
<tr>
<td>213</td>
<td>{([22, 227, 83, 189, 213], 60194)}</td>
</tr>
<tr>
<td>173</td>
<td>{([22, 103, 26, 304, 173], 57772)}</td>
</tr>
<tr>
<td>70</td>
<td>{([22, 70], 56372), ([22, 103, 26, 304, 173, 70], 31084), ([22, 202, 42, 70], 25288)}</td>
</tr>
<tr>
<td>13</td>
<td>{([22, 225, 60, 174, 13], 53120)}</td>
</tr>
<tr>
<td>235</td>
<td>{([22, 225, 60, 174, 235], 52305), ([22, 225, 60, 202, 42, 174, 235], 480)}</td>
</tr>
<tr>
<td>126</td>
<td>{([22, 191, 97, 222, 126], 49252)}</td>
</tr>
</tbody>
</table>

Table 3.1: Numerically encoded item and corresponding term

Table 3.2: Sample frequent patterns generated for DBpedia Ontology Types dataset. An item can be associated with multiple frequent patterns as seen for item 70.

### 3.1.1 Association Rule Mining

Frequent itemset mining is often associated with association rule mining, which involves generating association rules from the frequent itemset with constraints of minimal confidence (to determine if a rule is interesting or not). However, in this study, we do not require mining association rules using confidence values. Instead, we split the given database into
Table 3.3: Frequent patterns generated for the Geonames dataset. Each item is a pair of property and object \((p : o)\).

two disjoint databases, say \(A\) and \(B\), based on the frequent patterns. Those transactions which contain one or more of the top \(N\) frequent patterns are inserted into dataset \(A\) while the other transactions are inserted into dataset \(B\). Compression can be performed by creating a set of rules using top \(N\) frequent patterns and removing those triples from the dataset which can be inferred by applying rules to some other triples in the same dataset.

### 3.1.2 Multi-Dimensional Association Rules

Although association mining was originally studied for mining transactions for only one attribute (ex:Product), much research has been performed to extend it across multiple attributes [LFW05, LFH00, ZZ07, ZZJT00]. In this study, RDF datasets are viewed as multi-dimensional transaction databases by treating each property as an attribute and a subject as an identifier. Similar to intra-transaction and inter-transaction associations [LFH00], we define intra-property and inter-property associations for RDF datasets. Intra-property association refers to an association among different object values for a given property while inter-property association refers to association between multiple properties.
3.2 Rule Based Compression

Figure 3.3 depicts the high level overview of Rule Based Compression technique. We consider an RDF Graph $G$ containing $|G|$ non-duplicate triples. Lossless compression on graph $G$ can be obtained by splitting the given graph $G$ into an Active Graph, $G_A$, and a Dormant Graph, $G_D$, such that: $G = R(G_A) \cup G_D$ where $R$ represents the set of decompression rules to be applied to the active graph $G_A$ during decompression. $R(G_A)$ is the graph resulting from this application.

Since the compression is lossless, we have $|G| = |R(G_A)| + |G_D|$.

![Diagram of Rule Based Compression](image)

Figure 3.3: Rule Based Compression, $G = G_D \cup R(G_A)$

In the following sections, we provide brief introduction two RB compression algorithms - one using intra-property transactions and the other using inter-property transactions. In addition, we provide an algorithm for delta compression to deal with incremental compression when a set of triples needs to be added to existing compressed graphs. Specifically, we investigate how to

- generate a set of decompression rules, $R$
• decompose the graph \( G \) to \( G_A \) and \( G_D \), such that the requirements of RB compression holds true

• maximize the reduction in the number of triples

**Definition 4.** Let \( G \) be an RDF graph containing a set \( T \) of triples. An RB compression is a 3-tuple \( (G_A, G_D, R) \), where \( G_D \subset G \) is a dormant graph containing some triples \( T_D \subset T \), \( G_A \) is an active graph containing \( T_A \subset T - T_D \) triples and \( R \) is a set of decompression rules that is applied to \( G_A \) (denoted by \( R(G_A) \)) producing a graph containing exactly the set \( T - T_D \) of triples.

\( G_D \) is referred to as dormant since it remains unchanged during decompression (no rule can be applied to it during decompression).

### 3.2.1 Intra-property RB Compression

Algorithm 1 follows a divide and conquer approach. For each property in a graph \( G \), we create a new dataset and mine frequent patterns on this dataset. Transactions are created per subject within this dataset. Each transaction is a list of objects corresponding to a subject as shown in Figure 3.1. Using frequent patterns, a set of rules is generated for each property and later aggregated. Each rule contains a property \( p \), an object item \( k \), and a frequent pattern itemset \( v \) associated with \( k \). This rule will be used to expand compressed data given in \( G_A \) as follows:

\[
\forall x.\text{triple}(x, p, k) \rightarrow \bigwedge_{i=1}^{n} \text{triple}(x, p, v_i)
\]

where, \( v = v_1, v_2, ..., v_n \)

For illustration, here’s one such decompression rule we obtained during an experiment on DBpedia dataset:
∀x. triple(x, rdf:type, foaf:Person) → triple(x, rdf:type, schema:Person)
∧ triple(x, rdf:type, dbp:Person)
∧ triple(x, rdf:type, owl:Thing)

This triple is attached to the active graph $G_A$ so that all triples that can be inferred from it are removed. Other triples which cannot be inferred, are placed in dormant graph $G_D$. The process is repeated for all properties, appending results to the already existing rules $R$, active graph $G_A$ and dormant graph $G_D$.

### Algorithm 1 Intra-property RB compression

**Require:** $G$

1: \( R \leftarrow \phi, G_D \leftarrow \phi, G_A \leftarrow \phi \)
2: for each property, $p$ that occurs in $G$ do
3: create a transaction database $D$ from a set of intra-property transactions. Each transaction $\langle s, t \rangle$ contains a subject $s$ as identifier and $t$ a set of corresponding objects.
4: generate \( \{ (k, F_k) \} \) set of frequent patterns
5: for all $(k, F_k)$ do
6: select $v_k$ such that
7: \( \sigma(v_k) = \arg\max_v \{ \sigma(v) \mid v \text{ occurs in } F_k, |v| > 1 \} \)
8: \( R \leftarrow R \cup (k \rightarrow v_k) \) \( \triangleright \) add a new rule
9: end for
10: for each $\langle s, t \rangle \in D$ do
11: for each $(k \rightarrow v_k) \in R$ do
12: if $t \cap v_k = v_k$ then
13: \( G_A \leftarrow G_A \cup (s, p, k) \) \( \triangleright \) add single triple
14: \( t \leftarrow t - v_k \)
15: end if
16: end for
17: for each $o \in t$ do
18: \( G_D \leftarrow G_D \cup (s, p, o) \)
19: end for
20: end for
21: end for
3.2.2 Inter-property RB Compression

In Algorithm 2, we mine frequent patterns across different properties. Transactions used in this algorithm are created by generating a list of all possible pairs of properties and objects for each subject. Thus, each item of a transaction is a pair \((p : o)\). We follow a similar approach as before for generating frequent patterns and rules. Each rule contains a key pair \((p_k, o_k)\) and a corresponding frequent pattern \(v\) as a list of items \((p : o)\).

**Algorithm 2** Inter-property RB compression

```plaintext
Require: \(G\)
1: \(R \leftarrow \phi, G_D \leftarrow \phi, G_A \leftarrow \phi\)
2: create a transaction database \(D\) from a set of inter-property transactions. Each transaction, \((s, t)\) contains a subject \(s\) as identifier and \(t\) a set of \((p, o)\) items.
3: generate \(\{(k, F_k)\}\) set of frequent patterns
4: for all \((k, F_k)\) do
5: select \(v_k\) such that
6: \(\sigma(v_k) = \{ \text{argmax}_v \sigma(v) | v \text{ occurs in } F_k, |v| > 1 \}\)
7: \(R \leftarrow R \cup (k \rightarrow v_k)\) \(\triangleright\) add a new rule
8: end for
9: for each \((s, t) \in D\) do
10: for each \((k \rightarrow v_k) \in R\) do
11: if \(t \cap v_k = v_k\) then
12: \(G_A \leftarrow G_A \cup (s, p_k, o_k)\) \(\triangleright\) add single triple
13: \(t \leftarrow t - v_k\)
14: end if
15: end for
16: for each \((p, o) \in t\) do
17: \(G_D \leftarrow G_D \cup (s, p, o)\)
18: end for
19: end for
```

The procedure is similar to one described in 3.2.1 once frequent patterns and rules are generated.

\[
\forall x.\text{triple}(x, p_k, o_k) \rightarrow \bigwedge_{i=1}^{n} \text{triple}(x, p_i, o_i)
\]

For illustration, here’s one such decompression rule we obtained during an experiment on the Geonames dataset:
∀x.triple(x, geo:featureCode, geo:V.FRST) \rightarrow \text{triple}(x, \text{rdf:type}, \text{geo:Feature}) \land \text{triple}(x, \text{geo:featureClass}, \text{geo:V})

### 3.2.3 Optimal Frequent Patterns

In this section, we describe an optimal rule generation strategy for achieving better compression. In Algorithm 1 and Algorithm 2, we generate frequent patterns and keep only one frequent pattern $v$ per $k$. By selecting only one frequent pattern per item, it’s guaranteed that no circular reference or recursion occurs during decompression. As such, for any given triple in a compressed graph, only one rule can be applied.

The choice of $v$ for $k$ is determined based on whether $v$ has the maximum support. In this section, we present our findings for optimal $v$ pattern selection based on both support value and itemset length. To illustrate this finding, please consider a sample FP-Growth output obtained by mining one of the datasets as shown in Table 3.1 in section 3.1. If we look at frequent pattern sets for $k = 70$, we have:

1. $(v_1, \sigma_1) = ([22, 70], 56372)$
2. $(v_2, \sigma_2) = ([22, 103, 26, 304, 173, 70], 31084)$
3. $(v_3, \sigma_3) = ([22, 202, 42, 70], 25288)$

The following rule can be applied to select the optimal frequent pattern: select the pattern $v_i$ that maximizes $(|v_i| - 1) \times \sigma_i$. We call $(|v_i| - 1) \times \sigma_i$, denoted by $\rho(v_i)$, the **Redundant Triple Density**, signifying the total number of triples that can be removed by using a rule: $(k \rightarrow v_k)$. It is apparent that selecting $v_2$ during rule generation leads to higher compression than selecting $v_1$ or $v_3$.

We call $(|v_i| \times \sigma_i)$ the **Triple Density**signifying the total number of triples that are associated with this rule. Triple Density is also used to determine the top N Rules by sorting rules in descending order of the Triple Density corresponding to each rule.
3.2.4 Delta Compression

One of the important properties of RB compression is that incremental compression can be achieved on the fly without much computation. Let’s say, we consider an RDF graph $G$, which has undergone RB-Compression resulting in $G_A$ active graph, $G_D$ dormant graph and a set $R$ of decompression rules. If a new set of triples corresponding to a subject $s$, denoted by $\Delta T_s$, needs to be added to graph $G$, delta compression can be achieved by using the results from the last compression. Each delta compression updates the existing active and dormant graphs. Hence, there is no need for full RB-Compression every time a set of triples is added.

**Algorithm 3 Delta Compression**

**Require:** $G_A$, $G_D$, $R$, $\Delta T_s$
1: Extract all triples, $T_D$, corresponding to $s$ subject from $G_D$
2: $T \leftarrow T_D \cup \Delta T_s$
3: for all $t \in T$ do
4: if $R(t) \subseteq T$ then
5: $G_A \leftarrow G_A \cup t$ \hspace{1cm} $\triangleright$ insert into active graph
6: $T \leftarrow T - R(t)$
7: end if
8: end for
9: for all $t \in T$ do
10: $G_D \leftarrow G_D \cup t$ \hspace{1cm} $\triangleright$ insert into dormant graph
11: end for

Algorithm 3 provides a delta compression algorithm when $\Delta T_s$ needs to be added. The algorithm can be extended to include a set of subjects, $S$. It should be noted that we do not create new rules for a new set of triples. As such, the compressed version might not be optimal. A full compression is recommended if a large number of new triples needs to be added or if large number of delta compression have already been performed.

If a triple needs to be removed, an extra check needs to be performed to see if the removal violates any existing rules. Such removal might require moving some of the inferred triples from the active graph to the dormant graph.
3.3 Decompression

Decompression can be performed either sequentially or in parallel. Sequential decompression requires applying $R$ decomposition rules to triples in $G_A$ active graph and merging these inferred triples with the triples in $G_D$ dormant graph. Since each triple in a compressed graph can belong to at most one rule, its complexity is $O(|R| |G_A|)$. The number of rules is negligible compared to the number of triples in the active graph.

For parallel decompression, an active graph can be split into multiple smaller graphs so that each small dataset can perform decompression. This allows generation of inferred triples in parallel. Since rules are not ordered, inferred triples can be added to an uncompressed graph whenever they are generated. Finally, all triples of the dormant graph are merged into this uncompressed graph.

3.4 Evaluation

This section shows experimental results of the compression performed by our system. Our experiment is conducted on several linked open datasets as well as synthetic benchmark datasets of varying sizes. The smallest dataset consists of 130K triples while the largest dataset consists of 119 million triples.

3.4.1 RB Compression - Triple Reduction

Table 3.4.1 shows a comparison between the outputs of the two algorithms we discussed in Section 3.2 for nine different linked open datasets. The compression ratio, $r$ is defined as the ratio of the number of triples in the compressed dataset to that in the uncompressed dataset. It is evident from the results that compression based on inter-property frequent patterns is far better than compression using intra-property frequent patterns. Details including
the number of predicates and transactions derived during experiments are also included in the table. It can be seen that the best RB compression (inter-property) can remove more than 50% of triples for the CN datasets and DBpedia rdftype dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>triples (K)</th>
<th>predicate (K)</th>
<th>transaction (K)</th>
<th>compression ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>intra-property</td>
</tr>
<tr>
<td>Dog Food</td>
<td>130</td>
<td>132</td>
<td>12</td>
<td>0.98</td>
</tr>
<tr>
<td>CN 2012</td>
<td>137</td>
<td>26</td>
<td>14</td>
<td>0.82</td>
</tr>
<tr>
<td>ArchiveHub</td>
<td>431</td>
<td>141</td>
<td>51</td>
<td>0.92</td>
</tr>
<tr>
<td>Jamendo</td>
<td>1047</td>
<td>25</td>
<td>336</td>
<td>0.99</td>
</tr>
<tr>
<td>LinkedMdb</td>
<td>6147</td>
<td>222</td>
<td>694</td>
<td>0.97</td>
</tr>
<tr>
<td>rdftypes</td>
<td>9237</td>
<td>1</td>
<td>9237</td>
<td>0.19</td>
</tr>
<tr>
<td>RDF About</td>
<td>17188</td>
<td>108</td>
<td>3132</td>
<td>0.97</td>
</tr>
<tr>
<td>DBLP</td>
<td>46597</td>
<td>27</td>
<td>2840</td>
<td>0.96</td>
</tr>
<tr>
<td>Geonames</td>
<td>119416</td>
<td>26</td>
<td>7711</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table 3.4: Compression ratio for various linked open datasets

In addition to the compression ratio, we also measured: a) time it takes to perform RB compression and b) time it takes to perform full decompression. Figure 3.4 shows the compression and decompression times for various linked open datasets.

Figure 3.4: Compression and Decompression time for various linked open datasets
In general, RB compression time increases with the increase in triple size. However, if the total number of predicates in a dataset is very low, as in the case of DBpedia rdftypes dataset, compression time could be significantly lower. Decompression is faster by several order of magnitudes compared to the compression. This can be attributed to the fact that each triple is associated with a maximum of one rule and the number of rules are very few compared to the triple size. In addition, we apply rules only to triples in the Active Graph.

### 3.4.2 Comparison using compressed dataset size

In addition to evaluating our system based on triple count, we examine the compression based on the storage size of the compressed datasets and compare it against other compression systems. This is important since none of the existing compression systems has the ability to compress RDF datasets by removing triples. [FGMP10a] compared different universal compressors and found that bzip2\(^3\) is one of the best universal compressors. For this study, we compress the input dataset (in N-Triples format) and the resulting dataset using bzip2 and provide a quantitative comparison (see Table 3.4.2). An advantage of semantic compression such as RB Compression is that one can still apply syntactic compression (e.g. HDT) to the results. HDT [FMPG10a] achieves a greater compression for most of the datasets we experimented on. Such high performance can be attributed to its ability to take advantage of the highly skewed RDF data. Since any generic RDF dataset can be converted to HDT compact form, we ran HDT on the compressed dataset resulting from RB Compression. The experimental results are shown in Table 3.4.2. We see that this integration does not always lead to a better compression. This is due to the overhead of header and dictionary that HDT creates for both active and dormant dataset.

\[^3\text{http://bzip2.org}\]
### Table 3.5: Comparison of various compression techniques based on dataset size

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>compressed</th>
<th>compressed size using bzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>HDT</td>
</tr>
<tr>
<td>DogFood</td>
<td>23.4 MB</td>
<td>1.5 MB</td>
<td>1088 K</td>
</tr>
<tr>
<td>CN 2012</td>
<td>17.9 MB</td>
<td>488 K</td>
<td>164 K</td>
</tr>
<tr>
<td>Archive Hub</td>
<td>71.8 MB</td>
<td>2.5 MB</td>
<td>1.8 MB</td>
</tr>
<tr>
<td>Jamendo</td>
<td>143.9 MB</td>
<td>6 MB</td>
<td>4.4MB</td>
</tr>
<tr>
<td>LinkedMdb</td>
<td>850.3 MB</td>
<td>22 MB</td>
<td>16 MB</td>
</tr>
<tr>
<td>DBpedia rdftypes</td>
<td>1.2 GB</td>
<td>45 MB</td>
<td>11 MB</td>
</tr>
<tr>
<td>DBLP</td>
<td>7.5 GB</td>
<td>265 MB</td>
<td>201 MB</td>
</tr>
<tr>
<td>Geonames</td>
<td>13 GB</td>
<td>410 MB</td>
<td>304 MB</td>
</tr>
</tbody>
</table>

3.4.3 **RB Compression on Benchmark Dataset**

In this experiment, we ran RB Compression against one of the mainstream benchmark datasets, LUBM [GPH05a]. LUBM consists of a university domain ontology and provides a method for generating synthetic data of varying size.

Table 3.4.3 provides details on various LUBM datasets\(^4\) we used for the experiment. Not surprisingly, these results show that compression time on dataset increases with the increase in dataset size. However, the compression ratio remained nearly constant for all the synthetic dataset. Decompression time proved to be far lesser than the time required for compression as seen in Figure 3.5. It took only 200 seconds for the decompression of the LUBM 1000 dataset compared to 11029 second for the compression.

3.5 **Soundness and Completeness**

Although it should already be rather clear from our definitions and algorithms that our compression is *lossless* in the sense that we can recover all erased triples by using the

---

\(^4\)LUBM datasets created with index and seed set to 0.
Table 3.6: Compression ratio and time for various LUBM datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>triples (K)</th>
<th>transaction (K)</th>
<th>compression ratio</th>
<th>Time sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM 50</td>
<td>6654</td>
<td>1082</td>
<td>0.763</td>
<td>715</td>
</tr>
<tr>
<td>LUBM 100</td>
<td>13405</td>
<td>2179</td>
<td>0.757</td>
<td>1485</td>
</tr>
<tr>
<td>LUBM 200</td>
<td>26696</td>
<td>4341</td>
<td>0.757</td>
<td>2513</td>
</tr>
<tr>
<td>LUBM 500</td>
<td>66731</td>
<td>10847</td>
<td>0.757</td>
<td>6599</td>
</tr>
<tr>
<td>LUBM 1000</td>
<td>133573</td>
<td>21715</td>
<td>0.757</td>
<td>11029</td>
</tr>
</tbody>
</table>

Figure 3.5: Compression and Decompression time for various LUBM datasets

First of all, it is worth mentioning that we can recreate all erased triples by exhaustive forward-application of the rules—a fact that we could reasonably refer to as completeness of our approach. Our approach is also sound in the sense that only previously erased triples are created by application of the rules. I.e., our approach does not include an inductive component, but is rather restricted to detecting patterns which are explicitly and exactly represented in the dataset. Needless to say, the recreation of erased triples using a forward-chaining application of rules can be rephrased as using a deductive reasoning system as decompressor.

It is also worth noting that the rules which we introduce, which are essentially of the form \( \text{triple}(x, p, k) \rightarrow \text{triple}(x, p, v) \), can also be expressed in the OWL [HKP+09].
Web ontology Language. Indeed, a triple such as \((x, p, k)\) can be expressed in OWL, e.g., in the form\(^5\) \(k(x)\) if \(p\) is \texttt{rdf:type}, or in the form \(p(x, k)\) if \(p\) is a newly introduced property. The rule above then becomes \(k \sqsubseteq v\) for \(p\) being \texttt{rdf:type}, and it becomes \(\exists p.\{k\} \sqsubseteq \exists p.\{v\}\) in the case of the second example.

Because our compression rules are expressible in OWL, our approach to lossless compression amounts to the creation of schema knowledge which is completely faithful (in the sound and complete sense) to the underlying data. I.e., it amounts to the introduction of \textit{uncontroversial} schema knowledge to Linked Data sets. It is rather clear that this line of thinking opens up a plethora of exciting follow-up work, which we intend to pursue.

\(^5\)We use description logic notation for convenience, see [HKR09].
4 Contextual Linked Data Compression

In this chapter, we introduce a novel approach to compress RDF datasets by exploiting alignments [NS05] present across various datasets at both instance and schema level. The alignments expressed in Alignment API format contains the confidence value, which is used as a filter to drop the alignments below certain threshold. The threshold varies and is supplied by the user depending on the alignment systems and the application context. The equivalent terms based on alignments are represented with only one term in the compressed dataset. Hence, our approach results in a lossy compression.

4.1 Ontology Alignments

Ontology alignment refers to the task of identifying semantic similarities in different ontologies. The similarities can be between classes, entities or relations.

Given two ontologies $O_i$ and $O_j$, we can compute multiple mappings between the ontology terms, $t_i$ and $t_j$.

Alignment, $\mu$ is defined as $\mu = < t_i, t_j, r, s >$ where $r$ denotes the relationship and $s \in [0, 1]$ is the confidence score that the relationship holds in the mapping.
4.1.1 Schema Alignment

Datasets in Linked Data cater to different domains and thus contain a large number of domain specific ontologies. Different ontologies, whether belonging to the same domain or not, often share some similarities and can be aligned using classes and/or relations. Schema alignment can be performed using various approaches such as sense clustering [GdM09], instance similarity [IVDMSW07, WES08] and structural/lexical similarities[JMSK09]. Based on the schema alignment, the individual datasets can be rewritten using fewer schema terms. This leads to increased occurrences of same terms, resulting in a better compression.

4.1.2 Instance Matching

Datasets in Linked Data consists of many triples with similarity properties such as owl:sameAs, skos:closeMatch and skos:exactMatch. These similarity properties can be used to link entities and instances across multiple datasets. The task of identifying such similarities between entities is refereed to as Instance Matching. Different techniques are being used for instance matching such as exploiting the terminological structure, logical deduction and heuristics [SAS11]. Similar to the schema alignment, individual datasets can be rewritten using fewer mapped terms leading to a better compression.

It should be noted that alignment results vary greatly among different ontology matching systems (see [GDE+13]) for both schema alignment and instance matching. Some of these work best for one set of ontologies while performing poorly in a different set of ontologies. The alignments can differ even when manually performed among a group of experts for the same set of ontologies. For instance, conference track of OAEI provides the reference alignments\(^1\) with a confidence score of 1 (signifying exact match) for all mappings within a collection of ontologies describing the domain of organizing conferences. On the contrary, [CH14] introduced a new version of the Conference reference alignment

\(^1\)http://oaei.ontologymatching.org/2014/conference/data/reference-alignment.zip
for OAEI that includes confidence values reflecting expert disagreement on the matches.

## 4.2 Approach

In this section, we elaborate on the internals of our compression system. The main task involves identification of alignments across various datasets. The alignments can be manual or generated using existing Ontology matching systems.

Fig 5 represents the high level overview of our system. Given a set of input datasets, we first identify alignments present across these datasets. For this, we extract terms from each dataset and check for alignments with other participating datasets either manually or using automated ontology matching systems. This process can be skipped if the reference alignment is already generated for participating datasets, as in the case of OAEI Conference datasets. It should be noted that the alignments can be in both schema and instance level. The set of alignments are then consolidated by performing mapping to a set of *master* terms and pruning all mappings that have a confidence score below the threshold.

![Conceptual System Overview](image)

Figure 4.1: Conceptual System Overview

---

The resulting unique set of mappings, together with the original datasets, go through a transformation phase where all the datasets are merged and the equivalent terms are replaced with *master* terms. Fig. 4.2 shows two master items: ekawRegular_Paper and ekawResearch_Topic and corresponding consolidated alignments. Master items can be any one in the equivalent term set. Once the transformation is complete, the combined dataset is then passed to a Rule Based Compression (RBC) process. Details about RBC is explained in Chapter 3 above and in [JHD13]. The output from RBC can be represented in compact form using an adjacency list structure. The final compressed output consists of the equivalent mappings (required for decompression) and the output from RBC in compact form.

Figure 4.2: Grouping equivalent terms for ekaw#Regular_Paper and ekaw#Research_Topic using OAEI reference alignment.

Algorithm for the consolidation of alignments is listed in Algorithm 4. Given a threshold and a set of alignments, mappings with confidence score less than a threshold are pruned and a set of master items is generated. Each master item maps to a group of equivalent ontology terms. These master items are later used to rewrite the dataset to replace ontology terms with corresponding master item.
Algorithm 4 Consolidation of Alignments

Require: A Alignment set and $\theta$ threshold for alignments

1: Valid Mapping $M$ as $\langle k, V \rangle \leftarrow \emptyset$
2: Term Mapping $G \leftarrow \emptyset$
3: Set $S \leftarrow \emptyset$
4: MasterItem Mapping $I \leftarrow \emptyset$
5: for each mapping, $<e_1, e_2, r, s>$ that occurs in $A$ do
6:  if $r = 'equivalence'$ and $s \geq \theta$ then
7:  $M \leftarrow M \cup \langle e_1, V \cup e_1 \rangle$ \hspace{1cm} $\triangleright$ add a new valid mapping
8:  $M \leftarrow M \cup \langle e_2, V \cup e_2 \rangle$
9:  end if
10: end for
11: Grouping equivalent terms
12: for all $\langle k, V \rangle$ in $M$ do
13:  if $k \notin \text{keys}(G)$ and $k \notin S$ then
14:  $G \leftarrow G \cup \langle k, V_k \rangle$ \hspace{1cm} $\triangleright$ mark this $k$ as master item
15:  $S \leftarrow S \cup k$ \hspace{1cm} $\triangleright$ mark this $k$ as processed item
16:  for each $t \in V_k$ do \hspace{1cm} $\triangleright$ group all items in $V_k$ under $k$
17:  $G \leftarrow G \cup \langle k, V_t \rangle$ \hspace{1cm} $\triangleright$ $k$ maps to $V_k \cup V_t$
18:  $S \leftarrow S \cup t$ \hspace{1cm} $\triangleright$ mark this $t$ as processed item
19:  end for
20: end if
21: end for
22: One to One mapping with master item
23: for each $(k, V) \in G$ do
24:  for each $v \in V$ do 
25:  $I \leftarrow I \cup \langle v, k \rangle$ \hspace{1cm} $\triangleright$ map to master item
26:  end for
27: end for

4.3 Evaluation

For this work, we built a prototype, LinkIt, in JAVA to test the validity of our approach. We experimented using reference alignments from OAEI.
4.3.1 OAEI Conference Ontology

OAEI Conference track\(^3\) contains 16 ontologies (see Fig. 4.3) that deal with conference organization. The site also includes a reference alignment (Conference:V1) containing alignments between seven ontologies. All mappings have a confidence score of 1 (i.e. exact match). For the same set of ontologies, a new set of reference alignment (Conference:V2) is available from [CH14]. The alignments are manually created using inputs from group of experts and have a varying confidence score. In later sections, we show how we can take advantage of varying confidence score for achieving lossy compression.

<table>
<thead>
<tr>
<th>Name</th>
<th>Classes</th>
<th>Datatype Properties</th>
<th>Object Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ekaw</td>
<td>74</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>Sofsem</td>
<td>60</td>
<td>18</td>
<td>46</td>
</tr>
<tr>
<td>Sigkdd</td>
<td>49</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>Iasted</td>
<td>140</td>
<td>3</td>
<td>38</td>
</tr>
<tr>
<td>Micro</td>
<td>32</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>Confious</td>
<td>57</td>
<td>5</td>
<td>52</td>
</tr>
<tr>
<td>Pcs</td>
<td>23</td>
<td>14</td>
<td>24</td>
</tr>
<tr>
<td>OpenConf</td>
<td>62</td>
<td>21</td>
<td>24</td>
</tr>
<tr>
<td>ConfTool</td>
<td>38</td>
<td>23</td>
<td>13</td>
</tr>
<tr>
<td>Crs</td>
<td>14</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>Cmt</td>
<td>36</td>
<td>10</td>
<td>49</td>
</tr>
<tr>
<td>Cocus</td>
<td>55</td>
<td>0</td>
<td>35</td>
</tr>
<tr>
<td>Paperdyne</td>
<td>47</td>
<td>21</td>
<td>61</td>
</tr>
<tr>
<td>Edas</td>
<td>104</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>MyReview</td>
<td>39</td>
<td>17</td>
<td>49</td>
</tr>
<tr>
<td>Linklings</td>
<td>37</td>
<td>16</td>
<td>31</td>
</tr>
</tbody>
</table>

Figure 4.3: OAEI Conference Track Ontologies

\(^3\)http://oaei.ontologymatching.org/2014/conference/index.html
4.3.2 Dataset Generation

Since our primary purpose is to validate that RDF data can be compressed in presence of alignments, we need a set of ontologies, reference alignment for those ontologies and RDF data large enough to be tested. For the evaluation, we generated large size of synthetic RDF data using SyGENiA\(^4\) tool and a set of Conference ontologies and the reference ontologies available from OAEI\(^5\). Given a set of queries and an ontology, SyGENiA tool can automatically generate a large number of individuals. The set of queries that we use for generating RDF data is available from\(^6\). In order to test the compression against dataset of varying size, we created multiple queries and generated eight different dataset. The size of evaluation dataset size is shown in Table 4.4.

<table>
<thead>
<tr>
<th>ontology</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conference</td>
<td>113</td>
<td>261</td>
<td>257</td>
<td>123</td>
<td>195</td>
<td>213</td>
<td>113</td>
<td>727</td>
</tr>
<tr>
<td>confOf</td>
<td>107</td>
<td>152</td>
<td>149</td>
<td>77</td>
<td>137</td>
<td>129</td>
<td>98</td>
<td>546</td>
</tr>
<tr>
<td>iasted</td>
<td>84</td>
<td>161</td>
<td>157</td>
<td>74</td>
<td>129</td>
<td>108</td>
<td>84</td>
<td>540</td>
</tr>
<tr>
<td>sigkdd</td>
<td>98</td>
<td>158</td>
<td>146</td>
<td>92</td>
<td>137</td>
<td>126</td>
<td>88</td>
<td>390</td>
</tr>
<tr>
<td>cmt</td>
<td>67</td>
<td>149</td>
<td>140</td>
<td>79</td>
<td>97</td>
<td>99</td>
<td>56</td>
<td>658</td>
</tr>
<tr>
<td>edas</td>
<td>107</td>
<td>192</td>
<td>181</td>
<td>90</td>
<td>137</td>
<td>139</td>
<td>108</td>
<td>769</td>
</tr>
<tr>
<td>ekaw</td>
<td>94</td>
<td>181</td>
<td>177</td>
<td>63</td>
<td>146</td>
<td>147</td>
<td>92</td>
<td>704</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>670</strong></td>
<td><strong>1254</strong></td>
<td><strong>1207</strong></td>
<td><strong>598</strong></td>
<td><strong>978</strong></td>
<td><strong>961</strong></td>
<td><strong>639</strong></td>
<td><strong>4464</strong></td>
</tr>
</tbody>
</table>

Figure 4.4: Dataset size for various set of queries.

4.3.3 Varied Alignments and Compression

We evaluated two versions of Conference reference alignment available from OAEI and [CH14]. These reference alignments include 16 ontologies related to the conference organization and they are based upon the actual conference series and corresponding web

\(^4\)https://code.google.com/p/sygenia/
\(^5\)http://oaei.ontologymatching.org/2014/
\(^6\)http://bit.ly/1hgNaRv
The mappings in the Conference:V1 are all set to be exact match.

Figure 4.5 shows the difference in mapping for the same pair of items in the two versions of OAEI Conference reference alignments.

<table>
<thead>
<tr>
<th>Conference:V1</th>
<th>Conference:V2</th>
</tr>
</thead>
</table>
| <Cell>
  <entity1 :resource='conference#Topic'/>
  <entity2 :resource='ekaw#Research_Topic'/>
  <relation>=</relation>
  <measure>1.0</measure>
  </Cell> |
| <Cell>
  <entity1 :resource='conference#Organization'/>
  <entity2 :resource='ekaw#Organisation'/>
  <relation>=</relation>
  <measure>1.0</measure>
  </Cell> |
| <Cell>
  <entity1 :resource='conference#Conference_volume'/>
  <entity2 :resource='ekaw#Conference'/>
  <relation>=</relation>
  <measure>1</measure>
  </Cell> |
| <Cell>
  <entity1 :resource='conference#Topic'/>
  <entity2 :resource='ekaw#Research_Topic'/>
  <relation>=</relation>
  <measure>0.54</measure>
  </Cell> |
| <Cell>
  <entity1 :resource='conference#Organization'/>
  <entity2 :resource='ekaw#Organisation'/>
  <relation>=</relation>
  <measure>0.77</measure>
  </Cell> |
| <Cell>
  <entity1 :resource='conference#Conference_volume'/>
  <entity2 :resource='ekaw#Conference'/>
  <relation>=</relation>
  <measure>0.23</measure>
  </Cell> |

Figure 4.5: Varying alignment for same pair of items.

Figure 4.6 compares the distribution of valid mappings for various thresholds for both reference alignments. The number of mappings are generated after the consolidation of alignments. As expected, the number of mapping decreases with the increase of threshold in Conference:V2 reference alignment.

Furthermore, for the same set of datasets, various ontology matching systems can produce different set of alignments. Figure 4.7 shows a comparison of various alignment systems with varying number of equivalent terms for same threshold, as seen in the results of OAEI\(^8\). The alignments are generated for the same set of ontologies used in Conference:V1 and Conference:V2 reference alignments. As seen in Figure 4.7, some alignment systems

\(^7\)http://oaei.ontologymatching.org/2014/conference/index.html
\(^8\)http://oaei.ontologymatching.org/2014/conference/eval.html
such as RSDLWB and OMReasoner generate all alignments with a confidence score of 1, while others like LogMap and XMap generate alignments with varying confidence score.

Since the alignment is not one to one, we cannot recover the original data once compressed and hence the compression is lossy.

The evaluation result for varying alignments is shown in Figure 4.8 for one of the datasets which has original size of 670MB. The compressed size can be compared against
the output resulting from HDT alone which is 56MB.

<table>
<thead>
<tr>
<th>AlignmentSystem</th>
<th>Compressed size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conference:V1</td>
<td>51</td>
</tr>
<tr>
<td>Conference:V2</td>
<td>53</td>
</tr>
<tr>
<td>AML</td>
<td>53</td>
</tr>
<tr>
<td>Logmap</td>
<td>51</td>
</tr>
<tr>
<td>OmReasoner</td>
<td>52</td>
</tr>
<tr>
<td>Maasmtch</td>
<td>51</td>
</tr>
<tr>
<td>rsdlwb</td>
<td>53</td>
</tr>
<tr>
<td>xmap</td>
<td>53</td>
</tr>
</tbody>
</table>

Figure 4.8: Compressed size (in MB) against original size of 670MB

4.4 Discussion

In this work, we have explored lossy RDF compression, the area which has barely been researched in the semantic web. We limited alignments to only those with equivalence relationships so that a group of equivalent terms can be replaced by only one identifier. Although relationships such as subsumes and isSubsumed, can be used to infer additional triples, the study is limited to compression and reasoning on triples is beyond the scope of this study.

We observed that for same set of ontology terms, multiple alignment systems can produce varying confidence scores. Whether the alignments are manually created or automatically generated, we need to select a threshold that’s large enough to signify that the relationship holds true.

Given a positive threshold, $\theta < 1$, we can extract all mappings such that $s \geq \theta$ and treat them as valid mappings. Since these mappings are not exact matches and the choice of threshold can vary depending on the chosen alignment system, we refer to our compression technique as Contextual Lossy Compression.
5 Alignment based Linked Open Data Querying System

In this chapter, we present an Alignment based Linked Open Data SPARQL Querying System (ALOQUS) which allows users to effectively pose queries to the LOD cloud without having to know the representation structures or the links between its many datasets. We consider ALOQUS as an important step in information retrieval and knowledge discovery process as it automatically maps the user’s query to the relevant datasets (and concepts) using state of the art alignment methods; then executes the resulting query by querying each of the datasets separately; and finally merges the results into a single, complete answer. We perform a qualitative evaluation of ALOQUS on several real-world queries and demonstrate that ALOQUS allows users to effectively execute queries over the LOD cloud without a deep understanding of its datasets. We also compare ALOQUS with existing query systems for the LOD cloud to highlight the pros and cons of each approach.

5.1 Linked Open Data and Data Retrieval

Linked Open Data (LOD) contains a large and growing collection of interlinked public datasets represented using RDF and OWL. Concepts (and instances) in a each dataset are connected to (and hence can be reached from) related concepts (and instances) from other
datasets through semantic relationships such as \textit{owl:sameAs}. Hence, the LOD cloud is becoming the largest currently available structured knowledge base with data about music, movies, reviews, scientific publications, government information, geographical locations, medicine and many more. It has a potential for applicability in many AI-related tasks such as open domain question answering, knowledge discovery and Semantic Web. However, to take advantage of the enormously extensive structured data in the LOD cloud, one must be able to effectively pose queries to and retrieve answers from it.

SPARQL [PS\textsuperscript{+}06] has emerged as the de-facto query language for the Semantic Web community. It provides a mechanism to express constraints and facts to retrieve the information, and the triples matching those constraints are returned to the user. 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 ease querying from an infrastructural perspective, data contributors have provided public SPARQL endpoints to query the LOD cloud datasets.

### 5.2 Challenges

Finding the relevant and specific data in LOD cloud is a challenge as it requires users to understand various concepts and datasets prior to creating a query and running against multiple endpoints. For example, consider the query “\textit{Identify films, the nations where they were shot and the populations of these countries.}” Answering this query requires a 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 practical given the size (and continued growth) of the LOD cloud. Furthermore, issues such as schema heterogeneity and entity disambiguation identified in [JHY\textsuperscript{+}10] present profound challenges with respect to querying of the LOD cloud. Each of these data sources can be queried separately, most often
through an end point using the SPARQL query language [PS+06]. Looking for answers making use of information spanning over different data sets is a more challenging task as the mechanisms used internally to query datasets (database-like joins, query planning) cannot be easily generalized to this setting.

With respect to a systematic querying of the LOD cloud, we believe that the following challenges, some of which are identified previously in [JHY+10], make the process difficult and need to be addressed.

### 5.2.1 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.

### 5.2.2 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 many music related datasets such as Jamendo,\(^1\) MusicBrainz,\(^2\) and BBC Music. Even though the datasets belong to the same domain, they have been modeled differently depending on the creator. This is perfectly fine from a knowledge engineering perspective, but it makes querying of the LOD cloud difficult as it requires users to understand the heterogeneous schemas.

---

\(^1\)http://dbtune.org/jamendo
\(^2\)http://dbtune.org/musicbrainz/
5.2.3 Entity Co-reference

The purpose of entity co-reference is to determine if different resources refer to the same real world entity [SH10]. Often the LOD datasets have overlapping domains and tend to provide information about the same entity [GJM09]. The similarity is identified by using similarity properties such as “owl:sameAs” or “skos:exactMatch.” For instance, LinkedMdb provides information about the “Romeo & Juliet” movie and provides direct reference to DBpedia using the owl:sameAs property. However, there are cases where the two instances might not be directly connected but a path exists for such a co-reference as shown in Figure 5.1. Here, the Geonames resource for China is linked to the CIA Factbook concept and the DBpedia concept for China, using an “owl:sameAs” link from the NYTimes dataset. Finding results in scenarios which do not have a direct link is thus possible by traversing some common well-known similarity properties and retrieving information from multiple datasets.

5.3 Approach

This section introduces the approach used for data discovery, how we use mappings for constructing sub-queries, and the technique used for processing and integrating the results.

ALOQUS accepts SPARQL queries serialized by the user using concepts from an up-
per level ontology (the *primary* ontology for phrasing queries) such as PROTON [TKM05].

ALOQUS identifies the datasets for each concept and federates sub-queries to be executed on these datasets primarily using mappings between the upper level ontology and the LOD cloud datasets. ALOQUS consists of several steps to achieve this.

5.3.1 Automatic mapping between upper level Ontology and Ontologies used in LOD Datasets

To create an automatic mapping between the upper level ontology and ontologies used in LOD Datasets, we use the BLOOMS ontology alignment system [JHS+10, JYV+11]. The choice of BLOOMS over other ontology alignment systems such as [DGB06, GAP11, LTLL09] is mainly due to its higher precision and recall on LOD datasets, as shown in [JHS+10]. The mappings provided by BLOOMS are at the schema level and thus complement the existing mappings at the instance level provided by the LOD cloud. Thus, reusing upper level ontologies like PROTON and SUMO [NP01] 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. In addition to the automatically generated mappings, we use the existing mappings used in [DKSP10] and those already available on the web.\(^3\)

\(^4\) Our system is designed with pluggable architecture and hence can use output from any Alignment System that provides mappings in the Alignment API’s Alignment format [Euz04].

\(^3\)http://www4.wiwiss.fu-berlin.de/bizer/r2r/examples/ DBpediaToX.ttl
\(^4\)http://code.google.com/p/umbel/source/ browse/trunk/v100/External+Ontologies/
5.3.2 Identification and mapping of concepts in user defined queries to those in LOD Datasets

Using the mappings between an upper level ontology and other ontologies in the LOD datasets, the concepts specified in the query can be mapped to concepts of the LOD cloud datasets. Since the output of the alignment system, BLOOMS, is in the Alignment API format, the number of mappings can be restricted by providing a corresponding confidence threshold (the confidence value is a number between 0 and 1 that reflects the confidence of the system in the determined mapping). For instance, the mapping from “proton:school” to DBpedia for a threshold of 1 results in a mapping to “dbpedia:school” only, but for threshold of 0.9, we get additional mappings, for example to “dbpedia:EducationalInstitution.” BLOOMS suggest using a confidence value of 0.6 or higher but we found out that the number of mappings produced is often too many for our purpose so we restricted them to top k (variable) mappings that meet a threshold of 0.9.

5.3.3 Constructing Sub-queries

The concepts from the upper level ontology in a query are then substituted by mapped concepts to create multiple sub-queries. Each sub-query is created based on the concepts present in the corresponding datasets and taking cognizance of the fact, that some vocabularies such as FOAF, RDF and SIOC are reused by other datasets. Each of the sub-queries uses SPARQL CONSTRUCT (with upper level concepts in the graph template) instead of the SELECT query form to return an RDF graph containing triples with upper level concepts. The CONSTRUCT query form provides a mechanism to create new sets of triples, thereby making implicit LOD information explicit.
### 5.3.4 Execution of sub-Queries

For each sub-query, a graph is constructed by querying corresponding endpoints. For instance, a sub-query containing a statement with Music Ontology\(^5\) concepts is queried to both BBC Music\(^6\) and Jamendo endpoints. Source selection can be done either by specifying a local metadata file [QL08] or by sending a SPARQL ASK query for each triple pattern to every possible endpoint [SHH\(^+\)11]. For ALOQUS, we built a metadata file containing a list of endpoints, each mapped to ontologies used for the mapping. Information about vocabulary and endpoints are obtained from the CKAN directory.\(^7\) In addition, we consumed SPARQL services from Mondeca Labs’ LOV endpoints\(^8\) for vocabularies and endpoints. It should be noted that the returned graph contains triples with upper level concepts and LOD entities since upper level concepts are included in the CONSTRUCT graph template.

### 5.3.5 Determining entity co-references

The foundation of the LOD cloud is on the reuse of URIs across datasets, typically to assert similarity between entities or to link them. In order to search for entities similar to the variables of the queries created in the previous step, we use a crawling approach that detects the additional entities through owl:sameAs and skos:exactMatch. The crawling is required because two entities might not be directly connected but via other similar entities as exemplified in Section 5.2 above. A query used for fetching similar entities resembles the following.

```
SELECT ?sameAs ?property_var
WHERE
{ { { dbpedia:Hawaii owl:sameAs ?sameAs } union { ?sameAs owl:sameAs dbpedia:Hawaii } } }
```

---

\(^5\)http://musicontology.com/
\(^6\)http://api.talis.com/stores/bbc-backstage/services/sparql
\(^7\)http://thedatahub.org/
\(^8\)http://labs.mondeca.com/endpoint/lov/
A simple crawling approach used in ALOQUS is described below. For each entity retrieved from a sub-query, a new query is constructed using owl:sameAs and skos:exactMatch (see above) and then queried to multiple endpoints. Following an iterative approach, it fetches similar entities and inserts them into a Set. The final result for each entity is a unique list of similar entities which are then stored in a database under a unique identifier created on the fly (eg: http://www.knoesis.org/alosquuid). The creation of such a unique identifier greatly helps for querying in subsequent steps when join needs to be performed. We call them proxy identifiers and a set of similar entities corresponding to each proxy identifier a Similarity Set. The steps can be summarized as follows.

1. Get list of entities by executing a sub-query.

2. For each entity, construct a new query using owl:sameAs and skos:exactMatch (as shown above).

3. Query to an endpoint and fetch the similar entities.

4. Store the entities in a Similarity Set.

5. For each entity in a Similarity Set which has not yet been queried, repeat steps 2 to 4. This time, the endpoint will be different.

6. Merge the constructed sets if required.

In addition to our own crawling approach, we consume REST services from the sameAs.org website\(^9\) for getting equivalent URIs. It currently has over 100M URIs and returns back URIs which are co-referents for a given URI. It uses many predicates (ex:

\(^9\)http://www.sameas.org/
skos:exactMatch, cyc:similarTo) besides owl:sameAs to determine co-referent URIs from a variety of sources including DBpedia, NYTimes, UMBEL, OpenCyc and BBC Music. Using both of the mentioned approaches provides a larger (and hence more complete) set of similar entities and helps in identifying similar entities which do not have a direct link. We have presented a naive way to crawl for similar entities but the system gets better as we generate more proxy identifiers and add to our database. This step is an important step for ALOQUUS as it enables using common identifiers for join operations, if required in a query.

5.3.6 Transformation and local storage of RDF graphs

The RDF graphs returned by the execution of sub-queries are transformed into new RDF graphs by replacing the values of variables with the proxy identifiers created during the process of entity co-reference detection. The transformed graphs are then stored to an RDF store. In addition, the mappings between each proxy identifier to corresponding similar LOD entities are also stored. The inclusion of newly created proxy identifiers in a local RDF store is important because it eventually allows us to treat our RDF store as an independent dataset and thus to perform the join operation required for the queries.

5.3.7 Joining and Processing of results

With all the results from sub-queries now stored in the local RDF store, the next step is to perform an original query on the latter. It should be noted that join operations, if required in the query, would be automatically done since we have transformed all the triples to use proxy identifiers for the values of shared variables. The results can be considered final but the results include the values of variables represented in proxy identifiers. Since the mappings from proxy identifiers to values of variables returned from sub-queries are available in the datastore, all we need is to expand the result and present it to the user.
5.4 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
   ?film rdf:type protonu:Movie^{11}.
   ?nation protont:populationCount^{12} ?pop. }

2. By utilizing the BLOOMS mappings and getting the best alignment ($k = 1$) for each of the concepts, a set of sub-queries is generated where LOD cloud dataset specific concepts are substituted in lieu of upper level ontology concepts.

---

^{10}http://proton.semanticweb.org/2005/04/protonu#ofCountry
^{11}http://proton.semanticweb.org/2005/04/protonu#Movie
^{12}http://proton.semanticweb.org/2005/04/proton#PopulationCount
3. The subqueries are then executed in the corresponding end-points. Both the above sub-queries are transformed to use the SPARQL CONSTRUCT query form so that we get the graph instead of a table of results. It should be noted that the CONSTRUCT clause uses concepts from the upper level ontologies. For instance, the sub-query 2a is converted to

```
    ?film rdfs:label ?film_name. }
    ?film rdfs:label ?film_name. }
```

4. Some triples from the returned graphs (in Turtle format) are shown below. This includes triples with LOD entities and upper level concepts.

```
lmdb-film:11446 protonu:ofCountry lmdb-country:IN.
lmdb-film:11446 rdfs:label "Run".
lmdb-film:17091 protonu:ofCountry lmdb-country:LK.
lmdb-film:17091 rdfs:label "Getawaryo".
lmdb-film:16973 protonu:ofCountry lmdb-country:IN.
lmdb-film:16973 rdfs:label "Kabeela".
```
5. By looking at the above partial results, we can find that two results can be merged (treating dbpedia:India same as lmdb-country:IN). However, the lack of common identifiers keeps the triples from two results separate. The next step is to crawl and find out the similar entities. By using the entity co-reference detection process explained earlier, some of the similar entities from the similarity set of lmdb-country:IN and lmdb-country:LK include

http://data.linkedmdb.org/resource/country/IN
http://sws.geonames.org/1269750/
http://rdf.freebase.com/ns/m.03rk0
http://dbpedia.org/resource/India
http://data.nytimes.com/india_geo
http://dbtune.org/musicbrainz/resource/country/IN
http://umbel.org/umbel/ne/wikipedia/India
http://www.ontologyportal.org/SUMO.owl#India
http://www4.wiwiss.fu-berlin.de/factbook/resource/India

and

http://data.linkedmdb.org/resource/country/LK
http://rdf.freebase.com/ns/m.06m_5
http://dbpedia.org/resource/Sri_Lanka
http://data.nytimes.com/sri_lanka_geo
http://lexvo.org/id/iso3166/LK
http://linkedgeodata.org/triplify/node42431565
http://mpii.de/yago/resource/Sri_Lanka
http://psi.oasis-open.org/iso/3166/144
http://sw.opencyc.org/2008/06/10/concept/en/SriLanka

respectively.
6. The proxy identifiers and similarity sets are created at the same step resulting, e.g., in aloqus:2908ba82 and aloqus:9bc35ca1 identifiers for all the items in the similarity set of lmdb-country:LK and lmdb-country:IN, respectively.

7. The RDF graphs returned by the execution of sub-queries are transformed to include only the proxy identifiers for all the values of the variables that are shared among multiple statements in the original query. The variable *film* need not have the proxy identifiers but the *nation* should, since it is used in more than one statement. In essence, we are looking for common identifiers that would aid in the join operation.

```
lmdb-film:11446 rdfs:label "Run".
```

```
lmdb-film:17091 rdfs:label "Getawarayo".
```

```
```

```
```

```
aloqus:2908ba82 protont:populationCount 21324791.
```

```
aloqus:9bc35ca1 protont:populationCount 1210193422.
```

8. The transformed graphs are stored in a local RDF store and an original query is executed on it to fetch the results. The intermediate and final results are shown in Tables 5.1 and 5.2.

<table>
<thead>
<tr>
<th>film</th>
<th>name</th>
<th>nation</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>lmdb-film:17091</td>
<td>“Getawarayo”</td>
<td>aloqus:2908ba82</td>
<td>21324791</td>
</tr>
<tr>
<td>lmdb-film:16973</td>
<td>“Kabeela”</td>
<td>aloqus:9bc35ca1</td>
<td>1210193422</td>
</tr>
<tr>
<td>lmdb-film:11446</td>
<td>“Run”</td>
<td>aloqus:9bc35ca1</td>
<td>1210193422</td>
</tr>
</tbody>
</table>

Table 5.1: Result containing proxy identifiers
Table 5.2: Result containing LOD identifiers

<table>
<thead>
<tr>
<th>film</th>
<th>name</th>
<th>nation</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>lmdb-film:17091</td>
<td>“Getawayayo”</td>
<td>lmdb-country:LK</td>
<td>21324791</td>
</tr>
<tr>
<td>lmdb-film:16973</td>
<td>“Kabeela”</td>
<td>lmdb-country:IN</td>
<td>1210193422</td>
</tr>
<tr>
<td>lmdb-film:11446</td>
<td>“Run”</td>
<td>lmdb-country:IN</td>
<td>1210193422</td>
</tr>
<tr>
<td>lmdb-film:11446</td>
<td>“Run”</td>
<td>nytimes:india_geo</td>
<td>1210193422</td>
</tr>
</tbody>
</table>

5.5 Evaluation

As a proof-of-concept evaluation for our alignment based approach towards querying of Linked Open Data, an ALOQUS prototype has been implemented using the Jena\textsuperscript{13} Semantic Web Framework. The system takes a SPARQL query serialized by the user using concepts from the upper level ontology, and generates the appropriate mappings. For our purpose, we generated mappings between PROTON and various LOD ontologies including DBpedia, Music Ontology, LinkedMdb, the BBC Programme Ontology,\textsuperscript{14} Factbook\textsuperscript{15} and Semantic Web Corpus.\textsuperscript{16} These mappings are generated only once and additional mappings can be generated and added at any later time. ALOQUS then generates multiple sub-queries, executes them and crawls for co-referent URIs before merging the results and presenting the results to the user. The intermediate results are stored in a local TDB Store.\textsuperscript{17}

A standard measure for assessing the quality of querying systems are precision and recall. In our case, however, there does not exist any benchmark nor are there any baselines available for measuring these statistics partly because not much work has been done in alignment based query processing systems. Furthermore, the sheer size of the LOD cloud and its continuing growth makes it difficult to identify if all correct answers have been retrieved and reported. For these reasons, we present a test harness consisting of three different query types (discussed in Section 5.5.1) that can be used for evaluating ALOQUS.

\textsuperscript{13}http://jena.sourceforge.net/
\textsuperscript{14}http://www.bbc.co.uk/ontologies/programmes/2009-09-07.shtml
\textsuperscript{15}http://www.daml.org/2003/09/factbook/factbook-ont
\textsuperscript{16}http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf
\textsuperscript{17}http://openjena.org/TDB/
and similar systems that will be developed by researchers in our community in the future. We will propose a future evolution of this test harness through the Ontology Alignment Evaluation Initiative (OAEI).\(^{18}\)

We also performed a qualitative evaluation of our system by comparing it with DARQ \cite{QL08} and SQUIN \cite{HBF09}. Systems like Factforge \cite{BKO+11} are not used for comparison because they can be considered as working on a single dataset created by assembling multiple independent datasets. Our objective is to determine whether our system allows users to execute and retrieve answers to SPARQL queries over the LOD cloud without intimate knowledge of individual datasets and by using 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. We show that our reliance on BLOOMS, a state of the art alignment system, provides adequate answers to our queries.

### 5.5.1 Statement and Query Types

In this section, we introduce several terms for classifying queries that any alignment based querying system can be evaluated on with respect to a collection of datasets. To differentiate different query types, we introduce three types of query statements viz., *Akin Statement*, *Alien Statement* and *Allied Statement*.

A statement \(S\) occurring in a query \(Q\) is classified as an *Akin Statement* if all the predicates (concepts or properties) mentioned in the statement belong to the reference set of LOD ontologies. On the other hand, a query statement is an *Alien Statement* if none of the concepts and properties mentioned in the statement can be found in ontologies in the reference set (for instance, a statement containing terms from the upper level ontology only). An *Allied Statement* is one which has a combination of predicates, at least one existent and one non-existent in the reference set of ontologies. This type of query statement is of

\(^{18}\)http://oaei.ontologymatching.org
particular importance since the user has partial knowledge of the expected triples. The notion of Akin Statement generally refers to the connected statements that are already present in the reference datasets. Based on these statement types, the following query types are introduced:

- **Domestic Query**: A query containing only Akin Statements.
- **Foreign Query**: A query containing only Alien Statements.
- **Hybrid Query**: A query containing a combination of different statement types.

Each of the query types has a different level of complexity with respect to the required number of combinations of mappings, detection of equivalent URIs and the query federation. Domestic Queries do not need mappings and hence require only query federation and joins. Both Foreign and Hybrid queries involve predicate mappings in addition to federation and joins to fetch the results. Queries containing Alien statements can lead to a huge number of mappings and require both crawling and federation to a large number of endpoints. It should be noted that execution of Foreign queries within the reference datasets will always return an empty result set since the relevant concepts and properties do not occur in any triples in these datasets.

We further declare a set $V$ of vocabularies, whose appearance in the query statement should be ignored for classifying statement types. This flexibility is provided taking into consideration the fact that certain vocabularies such as RDF and FOAF have ubiquitous presence and are often required even when a user wants to use only upper level ontologies.

### 5.5.2 Queries and Results

For evaluation purposes, we created queries of different types which require information from multiple LOD datasets, and serialized them into SPARQL queries using concepts
The queries have been executed successfully by ALOQUS in a manner similar to Query 1 which was illustrated in Section 5.4. Query 1, of type Foreign, does not involve any concepts from LOD cloud datasets and the mentioned terms are properties or concepts from the upper level ontology. This involves the processing of results of queries on LOD datasets, which do not share a direct link in the LOD cloud. Thus, ALOQUS can unify answers even when sub-query answers are not directly connected to each other. Query 2, of type Domestic, has been obtained from the Semantic Web Dog Food website\(^{19}\) and does not require any mappings to be performed. Query 3 is another example of type Domestic but requires querying multiple datasets (Jamendo, Geonames) to get the results. Query

\(^{19}\)http://data.semanticweb.org/
4 (adopted from FactForge), of type Hybrid, contains concepts and properties from both the upper level ontology and from LOD datasets, and hence requires mappings for some property and concepts from the upper level ontology. Queries Q5 to Q8 are a few more Hybrid and Foreign queries. As can be seen from Table 5.3, ALOQUS can execute and process queries involving one or multiple datasets. Queries Q9 and Q10 show the extended capabilities of ALOQUS and will be discussed in Section 5.6.

Our results demonstrate that we are able to provide a mechanism to execute queries on the LOD cloud without relevant datasets’ concepts in the query. The ALOQUS 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 ALOQUS approach allows effective federation of SPARQL queries over the LOD cloud by using PROTON, 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.

### 5.5.3 Qualitative comparison with other tools

Two of the current systems, DARQ and SQUIN, which can partially answer some of the queries ALOQUS can, are compared on various metrics including query creation and entity co-reference detection as shown in Table 5.4. The queries were executed for ALOQUS. For other systems it is based on an understanding of the capabilities of the system. DARQ [QL08] 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 which to send triple patterns. Thus it requires the use of multiple queries to fetch results for queries of type Hybrid and Foreign. The absence of a direct link between different datasets often makes it impossible to fetch results for DARQ (queries similar to Q1). SQUIN allows LOD query answering
<table>
<thead>
<tr>
<th>Features</th>
<th>ALOQUS</th>
<th>DARQ</th>
<th>SQUIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach</td>
<td>Uses upper level ontology(PROTON) or any other ontology as primary ontology for query serialization and execution.</td>
<td>Requires formal description of datasets in the form of Service Description.</td>
<td>Requires an initial URI to execute queries.</td>
</tr>
<tr>
<td>Query Creation</td>
<td>Creates query corresponding to every mapping for a concept.</td>
<td>Creates queries only corresponding to the concepts mentioned in the query.</td>
<td>Creates queries only corresponding to the concepts mentioned in the query.</td>
</tr>
<tr>
<td>Failsafe</td>
<td>Executes all sub-queries for multiple mappings. Hence retrieves at least partial answers if a specific endpoint doesn’t work.</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Detect Entity co-references</td>
<td>Crawls and also consumes sameAs.org webservices.</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Result Processing</td>
<td>Query answers, retrieved from different datasets are merged and presented to user.</td>
<td>Retrieves answers from multiple dataset based on service description.</td>
<td>Retrieves answers from multiple dataset through link traversal.</td>
</tr>
<tr>
<td>Write queries using ontology not present in LOD</td>
<td>Yes</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Support for open-ended queries like ?s ?p ?o</td>
<td>Yes</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Result Storage for later Retrieval</td>
<td>Yes</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DESCRIBE Query Form</td>
<td>Yes</td>
<td>N/A</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 5.4: Comparison of LOD SPARQL Query Processing Systems
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 able to answer queries of both the Hybrid and Foreign types which include predicates not present in the existing datasets. Both DARQ and SQUIN are expected to fetch results for Domestic queries.

5.6 Discussion

Although many domain specific ontologies have been developed to facilitate the data integration, we still face challenges in assimilating information from multiple sources, which is essential for knowledge discovery. Our System, ALOQUS aims to seamlessly integrate data from multiple sources using upper ontology thereby enhancing the information retrieval and knowledge discovery in linked data.

Since ALOQUS is an alignment based querying system, there is no need to limit it to using only an upper level ontology as the primary ontology for phrasing the queries. This caters for cases where the user wants to query concepts that are not in the upper level ontology but exist in some LOD dataset, or if the user wants to use a different primary ontology such as DBpedia and use ALOQUS to get additional LOD data. A user also may have a proprietary ontology to be used for phrasing queries.

Since it is impossible to create one unique ontology that can map to every other LOD dataset, ALOQUS is designed to accommodate such alternative settings. The pluggable architecture of ALOQUS allows users to use any other upper ontology or LOD ontology as a primary ontology provided that a mapping can be generated (or provided) between the chosen primary ontology and other LOD ontologies.

While we presented an implementation that uses our state of the art alignment system, BLOOMS, it has a flexible architecture and can use any other alignment system that might
perform better in specific domains. One of the key strengths of ALOQUS architecture is that it enables automation of all the steps involved in query processing. ALOQUS is still a working prototype and lots of enhancement can be done with better optimization techniques. At present, the equivalent URI detection phase takes longer than the rest as a large number of crawling is performed for generating proxy identifiers and Similarity Sets.

The pluggable architecture, which enables the easy use of other primary ontologies and of other alignment systems and available mappings, means that ALOQUS can be modified for different purposes, and will gain in strength as further ontologies, mappings, and alignment systems become available. ALOQUS thus scales in the sense that it can easily improve as more data and tools for dealing with LOD datasets become available.
6 Synthetic Linked Data Generator

In this chapter, we present LinkGen, a synthetic linked data generator that can generate a large amount of RDF data based on certain statistical distribution. Data generation is platform independent, supports streaming mode and produces output in N-Triples and N-Quad format. Different sets of output can be generated using various configuration parameters and the outputs are reproducible. Unlike existing generators, our generator accepts any vocabulary and can supplement the output with noisy and inconsistent data. The generator has an option to interlink instances with real ones provided that the user supplies entities from real datasets. The tool is open source and available at GitHub\(^1\) under GNU License\(^2\).

6.1 Synthetic Linked Data

LOD datasets use a number of vocabularies to describe the group of related resources and relationships between them. According to [VAPVV15], Linked Open Vocabularies (LOV) dataset now consists of more than 500 vocabularies, 20,000 classes and almost 30,000 properties. The vocabularies are modeled using either RDF Schema (RDFS) or richer ontology languages such as OWL [BG04].

Linking enterprise data is also gaining popularity and industries are perceiving semantic technologies as a key contributor for effective information and knowledge management.

\(^1\)http://www.w3id.org/linkgen
\(^2\)https://opensource.org/licenses/GPL-3.0
One of the major obstacles for building a linked data application is generating a synthetic dataset to test against specific vocabularies.

Generating synthetic data is not a new concept. It has been widely used in database field for testing database design and software applications as well as database benchmarking and data masking [AKL11]. In the semantic web field, it has been primarily used for benchmarking Triplestores. Existing generators [MLANN11, SHLP09, BS08, GPH05b] are designed for specific use cases and work well with certain vocabularies but cannot be re-purposed for other vocabularies. LinkGen, on the other hand, can work with widely available vocabularies and can be used in multiple scenarios including: 1) Testing new vocabulary 2) Querying datasets 3) Diagnosing data inconsistencies 4) Evaluating performance of datasets 5) Testing Linked Data aggregators 6) Evaluating various compression methods

Creating synthetic datasets that closely resemble real world datasets is very important. Numerous studies including [FMPG10b, TDO07] found that URIs in real world linked datasets exhibit a power-law distribution. In order to automatically generate synthetic data that exhibit such power-law distribution, LinkGen employs random data generation based on various statistical distributions including Zipf’s Law

Real world linked datasets are by no means free of noise and redundancy. Linked Data quality and noise in Linked Data has been studied extensively in [PB14, WP14, PRMM11, ZRM15]. The noise can be in the form of invalid data, syntactic errors, inconsistent data and wrong statements. LinkGen provides some of these options to add noise in the synthetic dataset. LinkGen also has the option to specify the number of triples to generate. It aids in testing existing linked data compression methods such as [FMPG10b, JHD13] against varying database size and scenarios.

3To review Zipf’s and Pareto’s Law, see [Ada00]
6.2 Data Generator Features

In this section, we describe different concepts related to the data generator and provide details on how it works. At the core of data generation is a random data generator used for generating unique identifiers for each entity. In order to create different sets of output, LinkGen creates random data based on the seed value supplied by the user.

6.2.1 Entity Distribution

There are different statistical methods to generate and distribute entities in a dataset. LinkGen provides two statistical distribution techniques namely Gaussian distribution and Zipf’s power-law distribution. Example of Gaussian distribution includes those in real life phenomena such as heights of people, errors in measurement and marks on a test. Examples of Power-law distributions include the frequencies of words and frequencies of family names. [FMPG10b, TDO07] found that subject URIs in real world linked datasets exhibit a power-law distribution. LinkGen use zipf’s law as a default option for entity distribution. Figure 6.1 taken from [FMPG10b] shows the power-law distribution of subjects in a Wikipedia dataset.

6.2.2 Noisy Data

Noisy data plays a critical role in applications that aggregate data from multiple sources and those that deal with semi-structured and unstructured data [PB14]. LinkGen creates noisy data by:

- Adding inconsistent data, for instance writing two conflicting values for a given dataType property

- Adding triples with syntactic errors, ex: typos in subjectURI or rdfs:Label
Adding wrong statement by assigning invalid domain and range, ex: ns:PlaceInstance rdf:type ns:Person

Creating instances with no type information

Users can specify a combination of parameters for generating noisy data. All parameters related to noise are prefixed with noise.data text in the configuration file ex: noise.data.total and noise.data.num.notype. If the output is in N-Quads format, the noisy data are added to a separate named graph.

6.2.3 Inter-linking real world entities

LinkGen allows mapping real world entities with automatically generated entities. For this, the user has to supply a set of real world entities expressed in RDF format: <ns:entityuri> rdf:type <ns:subclass>. LinkGen will then interlink by using owl:sameAs triple, such as: <ns:entityuri>
Alternatively, the user can provide alignment file with various mappings and confidence score in the Alignment API format. This enables users to create a mixed dataset by combining synthetic dataset with the real dataset. This is important in scenarios where you would need to study the effect of adding new triples in current live dataset. Existing SPARQL queries can be slightly modified to fetch additional results from test dataset by adding owl:sameAs statement in the query.

6.2.4 Output Data and Streaming mode

LinkGen creates a VoID⁴ dump once the synthetic data is generated. VoID, the Vocabulary of Interlinked Datasets, is used to express metadata about RDF dataset and provides a number of properties for expressing numeric statistical characteristics about a dataset, such as the number of RDF triples, classes, properties or, the number of entities it describes.

LinkGen supports N-Triples and N-Quads format for output data. By default, the tool will save output to a file but it can be run in streaming mode, enabling users to pipe the output of RDF streams to other custom applications.

6.2.5 Config Parameters

There’s an array of configuration parameters available to create unique synthetic datasets. The output is reproducible so running LinkGen multiple times with same set of input parameters will yield same output. Most useful configuration parameters include: a) distribution type which can be gaussian or zipf and b) seed values for creating different datasets.

6.2.6 Data Generation Steps

The first step in data generation involves loading ontology and gathering statistics about all ontology components such as number of classes, datatype properties, object properties and

⁴https://www.w3.org/TR/void/
Properties for which domain and range are not defined. We also store the connectivity of each class and order the classes based on the frequency. Most connected class will lead to generation of larger number of corresponding entities.

The second step involves using statistical distribution to generate large number of entities and associating the weights for each one of them. Parameters for Zipf and Gaussian distribution are configurable and can be used to create different sets of output. For Zipf’s distribution, sample size is equal to the size of maximum number of triples to be generated. For Gaussian distribution, two parameters viz. mean and standard deviation are required.

Next step involves going through each class and generating synthetic triples for associated properties using weighted entities. For each entity, at least two triples are added to denote its type. They are: instance rdf:type Classs and, instance rdf:type owl:Thing. It should be noted that not all properties have well defined domain and range. For instance, in DBpedia, more than 600 properties including the ones in Table 6.1 have either missing domain or range information in the vocabulary. In such cases, RDF Semantics\(^\text{5}\) permits using any resources as a domain of the property. Similarly, the range can be any Literal or resource depending on whether the property is datatypeProperty or objectProperty.

For datatypeProperties which have range of XSD datatypes, we used a simple random generator to create literal values.

\(^\text{5}\)https://www.w3.org/TR/2000/CR-rdf-schema-20000327/
6.3 Evaluation

To evaluate our work, we generated varying number of synthetic datasets for two general purpose vocabularies: DBpedia\textsuperscript{6} and schema.org\textsuperscript{7}. For schema.org, we used an owl version available from TopBraid\textsuperscript{8}. We built LinkGen using Apache Jena\textsuperscript{9}, a widely used free and open source Java framework for building Semantic Web and Linked Data applications. At the current state, LinkGen supports only RDFS vocabularies. Although it can generate synthetic dataset for any vocabulary expressed in RDFS or OWL, it does not implement all class descriptions and property restrictions specified in the OWL ontology. Also, the support for blank nodes is not provided.

Table 6.2 shows the general characteristics of the dataset used for the experiment. For both DBpedia and Schema.org, the most connected classes were Person, Place and owl:Thing.

<table>
<thead>
<tr>
<th></th>
<th>DBpedia</th>
<th>Schema.org</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of distinct classes</td>
<td>147</td>
<td>158</td>
</tr>
<tr>
<td>Number of distinct properties</td>
<td>2891</td>
<td>1002</td>
</tr>
<tr>
<td>Number of distinct object properties</td>
<td>1734</td>
<td>463</td>
</tr>
<tr>
<td>Number of distinct data properties</td>
<td>1100</td>
<td>490</td>
</tr>
<tr>
<td>distinct properties without domain and/or range specification</td>
<td>685</td>
<td>11</td>
</tr>
</tbody>
</table>

Figure 6.2 is the performance chart depicting the total time taken to create synthetic datasets of varying size for both vocabularies. There’s a slight increase in time for DBpedia which may be due to the relatively high number of properties.

\textsuperscript{6}http://www.dbpedia.org
\textsuperscript{7}http://www.schema.org
\textsuperscript{8}http://topbraid.org/schema/
\textsuperscript{9}http://jena.apache.org/
6.4 Conclusion

In this chapter, we have introduced a multipurpose synthetic linked data generator. The system can be configured to generate various sets of output to test semantic web applications under different scenarios. This includes defining a statistical distribution type for instances, adding inconsistent and noisy data, and integrating real world entities. The system supports streaming mode which can be used for evaluating applications that deal with streaming data.
7 Related Work

In this chapter, we provide an overview of related work on compression and federated querying techniques. We also discuss few benchmark datasets and compare with our synthetic linked data generator.

7.1 RDF Compression

[FGMP10b] has explored various compression techniques for RDF datasets and observed that most RDF datasets are highly compressible due to its power-law distribution in term-frequencies, schema and resources. Work on frequent itemset mining [AIS93, HPYM04, LWZ08, ZEH10, ÖÀ10, ZPOL97] provides a foundation for our algorithms in Logical Linked Data Compression. [BC08] explored pattern mining based compression schemes for web graphs specifically designed to accommodate community queries. [VN11] used association rule mining techniques for generating ontology based on rdf:type statements.

In this section, we present some RDF serialization formats that take advantage of syntactic verbosity and skewed structure present in RDF datasets. General purpose compression algorithms such as Run-Length Encoding (RLE), bzip\(^1\), DEFLATE\(^2\), and Lempel-Ziv-Welch [ZL77] are excluded from the discussion. Various RDF representation formats including compact representations such as N-Triples are discussed in Section 2.2.

\(^1\)http://bzip.org/
\(^2\)http://www.faqs.org/rfcs/rfc1951.html
7.1.1 Adjacency List

RDF triples can be stored in a compact form by representing graph in adjacency lists. Turtle (and hence N3) allows such generalized adjacency lists for triples. Figure 7.3, taken from [FMPG+13], illustrates compact transformation of triples listed in Figure 7.1 using adjacency list. Figure 7.2 shows the dictionary for the terms present in these triples. Since \textit{dbr:Page2} appears in both \textit{subject} and \textit{object}, it has a precedence in dictionary over \textit{Page1}, \textit{Page4} and \textit{Page3} which appears either as \textit{subject} or \textit{object} only.

```turtle
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix dbp: <http://dbpedia.org/property/> .
dbr:page1 rdfs:label "Label1"@en.
dbr:page1 rdfs:label "Label2"@en.
dbr:page2 rdfs:label "Label3"@en.
dbr:page4 skos:broader dbr:page3
```

Figure 7.1: List of Triples

<table>
<thead>
<tr>
<th>ID</th>
<th>Dictionary</th>
<th>S-O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dbr:page2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>dbr:page1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>dbr:page4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>dbr:page3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>dbr:example1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Label1@en</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Label2@en</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Label3@en</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dbp:reference</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>rdfs:label</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>skos:broader</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>skos:subject</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.2: Dictionary encoding for terms in 7.1
**Definition 5.** An adjacency list $L$ of vertex $v \in V$ is a list that contains the vertices adjacent to $v$. The adjacency list representation of a graph $G$ consists of the adjacency lists of its vertices.

For example, the set of triples

$$\{(s, p_1, o_1), \ldots, (s, p_1, o_{1n_1}), \ldots, (s, p_2, o_{2n_2}), \ldots, (s, p_k, o_{kn_k})\}$$

can be written in adjacency list as

$$s \rightarrow [(p_1, ObjList_1), \ldots (p_k, ObjList_k)]$$

![Compact Transformation from ID-based triples using adjacency list](image)

**Figure 7.3:** Compact Transformation from ID-based triples using adjacency list

### 7.1.2 Bitmap

In the Bitmap format, the graph structure is indexed with two bit sequences, $B_p$ and $B_o$, for predicates and objects, in which 0-bits mark IDs in the corresponding $S_p$ or $S_o$ sequence, whereas 1-bits are used to mark the end of an adjacency list. This transformation is shown in Figure 7.4 taken from [FMPG+13]. Here, $Predicates=2,3,0,1,2,4,0,3,0$ evolves to the sequence $S_p=2,3,1,2,4,3$ and the bit sequence $B_p=001000101$ whereas, $Objects=6,0,2,0,3,0,4,5,0,1,0,2,0$ is reorganized in $S_o=6,2,3,4,5,1,2$ and $B_o=01010100101$.  

[ACZH10] introduced BitMat - a compressed bit-matrix structure for storing huge RDF graphs based on bitmap indexes.
7.1.3 HDT

[FMPG+13] introduced a compact representation format by decomposing an RDF dataset into three main parts: Header, Dictionary and Triples (hence, HDT). The Header is an optional component containing metadata about the data publication and is kept in plain text form. Dictionary organizes all the identifiers present in the RDF graph and is compressed by taking advantage of repeated URI prefixes and specific n-gram distributions in literals. The Triples component contains the structure of the data comprising the pure structure of the underlying graph and can be encoded in various formats including plain text, adjacency list and bitmap. Figure 7.5, taken from [FMPG+13], illustrates triple representation in various formats for list of triples shown in Figure 7.1.
7.2 Querying LOD

ALOQUS allows users to write query statements using concepts and properties not present in the LOD cloud. One area that is closely related to our work is query federation, which assumes that the user intimately knows concepts and datasets beforehand [QL08, HBF09, LW08, SHH+11]. [HL10] discusses a database perspective for querying Linked Data on the web including query federation, while [SHH+11] investigates optimizing techniques for federated query processing on Linked Data. ALOQUS also uses federation techniques to query distributed datasets once the sub-queries are generated. Systems like OBSERVER [MIKS00] have shown that the use of brokering across domain ontologies provides a scalable solution for accessing heterogeneous, distributed data repositories.

Work on ontology alignment and mapping [DFSB11, EMS+11a, DPS10, dMSP08] provides a foundation to our approach. Since ALOQUS uses an alignment system to generate sub-queries and then perform federation, any future improvement in state of the art alignment systems will also improve ALOQUS.

Another body of work which is related is work on upper level ontology creation. A number of well known upper level ontologies such as SUMO [NP01], Cyc [RL02], and DOLCE [GGM+02] are available [MCR06]. In the past various domain specific ontologies have been integrated with these upper level ontologies [O+07] driven by application specific needs. FactForge [BKO+11] uses mappings between the upper level ontology PROTON and other ontologies to build a compound dataset comprising some of the most popular datasets of the LOD Cloud, e.g., DBpedia, MusicBrainz, New York Times, Freebase, Geonames and Wordnet. Systems like PowerAqua [LFMS12] integrate ontology and natural language processing techniques for query answering.

Some of the existing endeavors on entity co-reference detection and resolution services [TD08, SH10, GJM09] are also related to our work as the join operation in ALOQUS is made possible by the detection of co-referent URIs.
7.3 Synthetic Linked Data Generator

To the best of our knowledge, LinkGen is the first work that generates synthetic linked dataset for any vocabulary with features such as statistical distribution, alignments and noisy data. Quite a few synthetic generators exist that have been developed for benchmarking RDF stores using specific vocabularies.

The Lehigh University Benchmark (LUBM) [GPH05b] consists of a data generator that produces repeatable and customizable synthetic dataset using Univ-Bench Ontology in the unit of a university. Different set of data can be generated by specifying the seed for random number generation, number of universities and the starting index of the universities.

Berlin SPARQL Benchmark (BSBM) [BS08] is built around an e-commerce use-case in which a set of products is offered by different vendors and consumers have posted reviews about products. BSBM constitutes a data generator that supports the creation of large datasets using number of products as the scale factor and can output in an RDF representation as well as relational representation.

SP²Bench [SHLP09] has a data generator for creating DBLP³-like RDF triples and mimics correlations between entities using power law distributions and growth curves. The Social Intelligence Benchmark (SIB) [PBE12] contains an S3G2 (Scalable Structure-correlated Social Graph Generator) that creates a synthetic social graph with correlations. Tontogen⁴ is a protege-plugin that can create synthetic dataset using a uniform distribution of instances for relationships. WatDiv⁵ and Sygenia⁶ are two other tools that can generate data based on user supplied queries. There has been some work on OWL TBox generators, e.g. [OVDT⁺08] and [BHSS11]:

As noted above, none of the existing generators are suitable for creating synthetic data

---
³http://dblp.uni-trier.de/db/
⁴http://lsdis.cs.uga.edu/projects/semdis/tontogen/
⁵http://dsg.uwaterloo.ca/watdiv/download
⁶https://sourceforge.net/projects/sygenia/
for different vocabularies. They are tied to specific vocabularies and have a little or no option to configure the output in regards to data distribution, noise and alignments.
8 Conclusion

In this chapter, we summarize the contributions of this dissertation and provide an outlook to possible future work.

8.1 Alignments and RDF compression

The success of linked data has resulted in a large amount of data being generated in a standard RDF format. Various techniques have been explored to generate a compressed version of RDF datasets for storage, archival and transmission purpose. However, these compression techniques are designed to compress a given dataset through compact representation and only removes syntactic and structural redundancies. The work presented here identified the logical and contextual components of linked data and leveraged them to achieve both lossless and lossy compression respectively.

In Chapter 3 we showed a lossless compression technique called Rule Based Compression that efficiently compresses RDF datasets using logical rules. The key idea is to split the original dataset into two disjoint datasets A and B, such that dataset A adheres to certain logical rules while B does not. Dataset A can be compressed since we can prune those triples that can be inferred by applying rules on some other triples in the same dataset. We have provided two algorithms based on frequent pattern mining to demonstrate the compression capability of our rule based compression. Experimental results show that in some datasets, RB Compression can remove more than half the triples without losing
data integrity. The set of logical rules can be considered as an alignment with n:m (many-to-many) cardinality.

In Chapter 4, we proposed a lossy compression technique that exploits alignments present across various datasets at schema and instance level. Our experiments on varied alignments using reference alignments from OAEI reveal that the alignments vary greatly depending on the alignment systems. For this reason, our system expects the user to provide a threshold value which is used to filter the alignments. Since the selection of threshold depends on the alignment system and/or the application context, we refer our lossy compression technique as contextual. To the best of our knowledge, this is the first study exploring the lossy compression of RDF dataset.

8.2 Alignments and Query Answering

In Chapter 5, we proposed a novel approach that allows querying of Linked Data without requiring that the user have intimate knowledge of individual datasets and interconnecting relationships. The basic idea of our approach is to make use of ontology alignment systems for querying. Our system supports writing queries using just an upper level ontology (e.g., PROTON) or cross-domain ontologies (e.g., DBpedia) or any other ontology as the primary ontology for expressing queries. Our methodology allows automatic retrieval and merging of results for queries that involve resources indirectly linked in the LOD cloud. Using this approach, we are able to answer queries which cannot be answered by state of the art systems for LOD query processing. With our initial test harness and sample queries, we hope that our community will develop a resource for evaluating future efforts in alignment-based querying systems.
8.3 Alignments and Synthetic Data Generation

In Chapter 6, we presented LinkGen, a multipurpose synthetic linked data generator that can automatically generate synthetic datasets without having to understand the schema. It tries to mimic the real world dataset by using power-law distribution and infiltrating the dataset with noisy data. The tool also supports generating a dataset in stream mode, which enables developers test their tools against streaming RDF data. By generating a large amount of RDF data that can be repeated and reproduced, it can aid in testing the performance of various applications that deal with querying, storage, visualization, compression and reporting. By incorporating alignments in the data generation process, we are able to interlink the real world entities with the randomly generated entities. Furthermore, multiple sets of test dataset can be generated based on the user supplied alignment file and the threshold.

8.4 Future Work

There are several directions in which future work can be pursued. The compression techniques presented in Chapter 3 and 4 can be extended to support querying over the compressed dataset. The Delta compression can be improved to measure the impact of addition and deletion of multiple triples and compare the fully compressed dataset against the incrementally compressed dataset. Since we have introduced both lossless and lossy compression, a qualitative study can be performed to determine the scenarios where lossy compression would make more sense. Logical rules derived during RB compression can be studied to support the extension of the dataset and/or creation of new schema. It should be noted that the datasets we used are modeled differently and hence some are highly structured than the others. For instance, GeoNames, Semantic Dog Food and DBLP are highly structured where as DBpedia is not. Further analysis is required to understand how com-
pression is affected by how well the dataset is structured.

While the contributions on alignment based querying system provide a novel querying approach for LOD, there is also a lot of room for improvement. Given the fact that our method depends on one of the currently available alignment systems, ALOQUS has limitations that stem from the limitations of BLOOMS, our chosen alignment system. Present day alignment systems try to find direct mappings between two different concepts. However, there are cases where the two concepts might not align directly but only if there is a chain of mappings as exemplified in the R2R Framework.\(^1\) Such mapping chains are currently not supported in ALOQUS. We believe that building a better alignment system is important and that alignment based querying systems like ALOQUS will greatly help users in writing queries without specifying exact relations and knowing datasets beforehand. Future work can include using ALOQUS together with entity summarization [CGQ08, GTS15] for exploring and visualizing entities. ALOQUS can also make use of VoID statistics [AH09] for source selection.

On generation of synthetic data, there are some ample opportunities for extending especially using some machine learning technique. This includes learning distributions from datasets like DBpedia, learning join probability distributions (e.g. if x is entailed to be a class of city, probability of having one population triple is 0.9) and learning RDF Shapes. It might also be useful to provide different kinds of distributions for different properties and classes. The synthetic dataset can be extended to automatically extract the instances from LOD cloud if the ontology contains the classes that already exist in the Linked Open Vocabularies.

\(^1\)http://www4.wiwiss.fu-berlin.de/bizer/r2r/
Bibliography


[QL08] Bastian Quilitz and Ulf Leser. Querying Distributed RDF Data Sources with SPARQL. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings, volume 5021 of


