University of Trento
University of Bergamo
University of Brescia
University of Padova
University of Trieste
University of Udine
University IUAV of Venezia

Md. Rashedul Hasan

Semantic Aware Representing and Intelligent Processing of Information in an Experimental domain: the Seismic Engineering Research Case

Tutor: Prof. Oreste S. Bursi

Co-Tutor: Feroz Farazi

April, 2015

UNIVERSITY OF TRENTO

Ph. D. in Engineering of Civil, Environmental and Mechanical
Cycle: XXVII

Head of Department: Prof. Marco Tubino

Head of the Doctoral School: Prof. Paolo Scardi

Final Examination: 24 / 04 / 2015

Board of Examiners

Prof. Claudio Di Prisco (Politecnico di Milano)

Prof. Ernesto D'Avanzo (Università di Salerno)

Prof. Enrico Canuto (Politecnico di Torino)

Abstract

Seismic Engineering research projects' experiments generate an enormous amount of data that would benefit researchers and experimentalists of the community if could be shared with their semantics. Semantics is the meaning of a data element and a term alike. For example, the semantics of the term experiment is a scientific research performed to conduct a controlled test or investigation. Ontology is a key technique by which one can annotate semantics and provide a common, comprehensible foundation for the resources on the Semantic Web. The development of the domain ontology requires expertise both in the domain to model as well as in the ontology development. This means that people from very different backgrounds, such as Seismic Engineering and Computer Science should be involved in the process of creating ontology. With the invention of the Semantic Web, computing paradigm is experiencing a shift from databases to Knowledge Bases (KBs), in which ontologies play a major role in enabling reasoning power that can make implicit facts explicit to produce better results for users. To enable an ontology and a dataset automatically exploring the relevant ontology and datasets from the external sources, these can be linked to the Linked Open Data (LOD) cloud, which is an online repository of a large amount of interconnected datasets published in RDF. Throughout the past few decades, database technologies have been advancing continuously and showing their potential in dealing with large collection of data, but they were not originally designed to deal with the semantics of data. Managing data with the Semantic Web tools offers a number of advantages over database tools, including classifying, matching, mapping and querying data. Hence we translate our database based system that was managing the data of Seismic Engineering research projects and experiments into KB-based system. In addition, we also link our ontology and datasets to the LOD cloud.

In this thesis, we have been working to address the following issues. To the best of knowledge the Semantic Web still lacks the ontology that can be used for representing information related to Seismic Engineering research projects and experiments. Publishing vocabulary in this domain has largely been overlooked and no suitable vocabulary is yet developed in this very domain to model data in RDF. The vocabulary is an essential component that can provide logistics to a data engineer when modeling data in RDF to include them in the LOD cloud. Ontology integration is another challenge that we had to tackle. To manage the data of a specific field of interest, domain specific ontologies provide essential support. However, they alone can hardly be sufficient to assign meaning also to the generic terms that often appear in a data source. That necessitates the use of the integrated knowledge of the generic ontology and the domain specific one.

To address the aforementioned issues, this thesis presents the development of a Seismic Engineering Research Projects and Experiments Ontology (SEPREMO) with a focus on the management of research projects and experiments. We have used DERA methodology for ontology development. The developed ontology was evaluated by a number of domain experts. Data originating from scientific experiments such as cyclic and pseudodynamic tests were also published in RDF. We exploited the power of Semantic Web technologies, namely Jena, Virtuoso and VirtGraph tools in order to publish, storage and manage RDF data, respectively. Finally, a system was developed with the full integration of ontology, experimental data and tools, to evaluate the effectiveness of the KB-based approach; it yielded favorable outcomes. For ontology integration with WordNet, we implemented a

semi-automatic facet based algorithm. We also present an approach for publishing both the ontology and the experimental data into the LOD Cloud. In order to model the concepts complementing the vocabulary that we need for the experimental data representation, we suitably extended the SEPREMO ontology. Moreover, the work focuses on RDF data sets interlinking technique by aligning concepts and entities scattered over the cloud.

ACKNOWLEDGEMENTS

In the first place I would like to record my gratitude to Professor Oreste S. Bursi for his supervision, advice, and guidance from the very early stage of this research as well for giving me extraordinary experiences throughout the work. Above all and the most needed, he provided me unflinching encouragement and support in various ways.

Special thanks go to my co-supervisor, Dr. Feroz Farazi, for his guidance, insightful discussion, and encouragement throughout the development of this thesis.

I am also thankful to The University of Trento for granting me the Ph.D. fellowship to commence this thesis and to carry out the necessary research work.

I would also like to thank all my friends, in particular, Dr. Md. Shahin Reza, for his encouragement during this challenging time and my PhD.

Finally, but not lastly, my special and heartfelt thanks to my parents, my family and my wife, Mishu, for supporting me and encouraging me with their best wishes.

CONTRIBUTIONS and PUBLICATIONS

The main contributions of this thesis are summarized as follows:

- An ontology based system for the Seismic Engineering domain is introduced to provide a mechanism to manage information and semantics thereof that can make systems semantically interoperable, and as such can exchange and share data.
- It develops faceted ontology in the Seismic Engineering domain.
- It provides an overview of the Semantic Web languages in order to identify a suitable language for representing faceted ontologies.
- Representing SEPREMO as an RDF graph that can help understanding the relationship between different concepts.
- The integration of the SEPREMO ontology with WordNet.
- To realize the theoretical concepts into practical systems and to make the results of this thesis accessible to the user, a number of tools have been implemented: an ontology browser with the possible support for searching, editing and visualizing both the ontology and experimental data.
- It provides an overview of how to deal high-volume, high-velocity and high-variety of information.
- It developed an ontology matching algorithm that potentially contributes to resolve the data integration and interoperability issue. The proposed algorithm is implemented so that users can select an RDF file to find the correspondences on the LOD cloud.
- An analysis of the computational complexity of the matching algorithm is also provided.
- It has developed a lightweight semantic search platform using Apache Lucene that supports user for searching documents by keywords.

As a result of the work conducted in this thesis, the following publications have been produced:

Journal Publication:

Hasan, M.R., Farazi, F., BURSI, O.S., Reza, M.S., D'Avanzo E. "A semantic-aware data management system for seismic engineering research projects and experiments", International Journal for Advanced Research in Artificial Intelligence, 2015, (in print).

Conferences:

Hasan M. R., Farazi F, Bursi OS, Reza M.S. Towards Publishing Earthquake Engineering Experimental Data As Part Of Linked Open Data Cloud. The 8th INSPIRE Conference, Aalborg, Denmark, 16-20 June 2014.

Hasan M. R., Farazi F., Bursi O. S., Reza M. S. A faceted lightweight ontology for earthquake engineering research projects and experiments. SERIES Concluding Workshop - Joint with US-NEES, Earthquake Engineering Research Infrastructures, JRC-Ispra, May 28-30, 2013.

Hasan M. R., Farazi F., Bursi O. S., Reza M. S. A Semantic Technology-Based Information Management System for Earthquake Engineering Projects and Experiments. 5th International Conference on Advances in Experimental Structural Engineering, Taipei, Taiwan, November 8-9, 2013.

A. Bosi, I. Kotinas, I. Lamata Martínez, S. Bousias, J.L. Chazelas, M. Dietz, Hasan M. R., S.P.G. Madabhusi, A. Prota, T. Blakeborough, P. Pegon. The SERIES Virtual Database: Exchange Data Format and local/central databases. SERIES Concluding Workshop - Joint with US-NEES, Earthquake Engineering Research Infrastructures, JRC-Ispra, May 28-30, 2013.

CONTENTS

1 INTRODUCTION **1**

1.1 The Context 1

1.2 The Problem 3

1.3 Solution 5

1.4 Structure of the Thesis 6

2 Background Studies **9**

2.1 Introduction 9

2.2 Exchange Data Format 10

2.3 System Architecture 14

2.3.1 Local site management of RELUIS Database 14

2.4 Distributed Database Architecture 15

2.5 Conclusion 18

3 State of the Art **19**

3.1 Introduction 19

3.2 Ontology 20

3.2.1 Ontology Design and development 24

3.2.2 Ontology Representation 27

3.2.3 Ontology Matching Techniques 30

3.2.4 Query Formulation and Answering 34

3.3 The Semantic Web 35

3.4 Data Science 38

3.4.1 R Statistical Tools 39

3.4.2 MatLab 40

3.4.3 Hadoop . . . 40
3.4.4 Open Refine . . . 42
3.4.5 Apache Spark . . . 43
3.5 Conclusion . . . 44

4 Ontology Development **47**
4.1 Introduction . . . 47
4.2 Ontology Development Methodology . . . 48
4.3 Ontology Representation . . . 51
4.4 Ontology Integration . . . 52
4.5 Ontology Mapping . . . 54
4.6 Ontology Alignment . . . 56
4.7 Ontology Evolution . . . 59
4.8 Conclusion . . . 62

5 Ontology Publishing **63**
5.1 Introduction . . . 63
5.2 Vocabularies . . . 63
5.3 Web Ontology Languages . . . 67
5.3.1 Resource Description Language (RDF) . . . 67
5.3.2 RDF Schema . . . 70
5.3.3 Web Ontology Language (OWL) . . . 72
5.4 Conclusion . . . 77

6 Linking RDF Datasets **79**
6.1 Introduction . . . 79
6.2 Linked data background and sources . . . 81
6.3 Ontologies alignment using Linked Data . . . 83
6.4 Ontologies alignment using Linked Data . . . 84
6.4.1 Algorithm . . . 84
6.4.2 System Design . . . 87
6.5 Evaluation . . . 87
6.6 Conclusion . . . 88

7 Intelligent Query Answering **91**
7.1 Introduction . . . 91
7.2 Document Collection . . . 92
7.3 Indexing and Searching . . . 92
7.3.1 Apache Lucene . . . 92
7.3.2 Apache Solr . . . 97
7.4 Output . . . 100
7.5 Conclusion . . . 100

8 Result **101**
8.1 Introduction . . . 101
8.2 Ontology Development . . . 102
8.2.1 Approach . . . 102
8.2.2 Experimental Data Collection . . . 103
8.2.3 Experimental Setup . . . 104
8.2.4 Controlled Experiment . . . 106
8.2.5 RDF Liking Datasets . . . 112
8.2.6 Intelligent Query System . . . 113
8.3 RELUIS Database . . . 115
8.4 Conclusion . . . 122

9 Conclusions **125**
9.1 Thesis Summary . . . 125
9.2 Limitations . . . 128
9.3 Future Work . . . 128
9.4 Conclusions . . . 129

Appendix A Seismic Engineering Research Projects and Experiments Management Ontology Facet **140**

LIST OF FIGURES

2.1 Data hierarchy 11
2.2 Example of Signal Table 12
2.3 Project, Sensor level in MySQL 15
2.4 RELUIS distributed database 15

3.1 RDF Triple 28
3.2 Matching algorithm 33
3.3 Semantic Web Architecture (Berners-Lee et al., 2001) 36

4.1 The Device and Experiment Facets 50
4.2 SEPREMO RDF graph 52
4.3 NEES Thesaurus 53
4.4 After applying the removal of the concept Vibrate. 60
4.5 Inclusion of the shaker-based test and hammer-based test in the experiment facet 61

5.1 A subset of the entity class concepts of the SEPREMO 65
5.2 A partial list of attribute concepts added to the SEPREMO 66
5.3 Excel view of SEPREMO 68
5.4 RDF describes the Resources 69
5.5 A snippet of the experimental data represented in RDF 70
5.6 Modeling Components of RDF schema from (Brickley et al. , 2004) . . 71
5.7 RDFS describes the Resources ”Hammer” and ”Damper” 71
5.8 Attribute presents in RDF 72
5.9 A fragment of OWL ontology. 74
5.10 A fragment of OWL ontology. 75

6.1 Linked open Data Cloud (Tim Berners-Lee , 2006) 80
6.2 The basic structure of a Semantic matcher algorithm. 85
6.3 System Architecture 87

7.1 Apache Lucene System Architecture 95
7.2 Intelligent query answering system architecture 96
7.3 The architecture of Solr (Yonik , 2006) 98
7.4 Figure: Solr Admin Panel 99

8.1 Ontology based development Approach 102
8.2 Experimental set-up of a piping system tested under earthquake loading (Reza et al. , 2013). 103
8.3 Ontology Integration and Population to KB 104
8.4 KB-based System Architecture 105
8.5 Synonymus relationship of Test. 107
8.6 Transitive Relationship of Device 108
8.7 Sparql Endpoint 108
8.8 Circle pack layout for ontology visualization 110
8.9 A snippet of the experimental data represented in RDF 111
8.10 Load-displacement characteristics curve drawn with the data provided in Figure 8.9 111
8.11 Web Interface for RDF Data sets Linking 112
8.12 WordNet RDF dataset interlink with DBpedia 113
8.13 SEPREMO RDF datasets interlink with DBpedia. 113
8.14 Intelligent Query System 114
8.15 The organization of the views of RELUIS application 115
8.16 Login Page 116
8.17 Add New Project 117
8.18 List of Projects 118
8.19 Navigation at the Central Site: project level 118
8.20 Specimen level: expanded view of specimen characteristics 119
8.21 Experiment level: tests have been performed 119

8.22 Signal level: each signal is delivered together with data regarding its units, the nature of the signal 120
8.23 List of Users 120
8.24 Search User Page 121
8.25 Profile Logo Change 121

LIST OF TABLES

3.1 The Hadoop Family 41

5.1 A partial list of relation concepts added to the SEPREMO 67

6.1 Performance of Semantic matcher 88

8.1 Statistics about SEPREMO ontology 106

CHAPTER 1

INTRODUCTION

1.1 The Context

The Semantic Web was designed to be the ground of meaning-wise interconnected, logically consistent, immediately updateable and machine processable data elements. These data elements can come from the original Web as well as from other sources ranging from universities and research centers to private and public organizations. Until the middle of the last decade people were barely publishing data on the Semantic Web because of the lack of skill for generating data and the deficiency of the easy to use tools for converting data into required logical formalisms. Moreover, tools which were already in place could hardly show their potential in dealing with large amount of data.

Since the advent of the Linked Open Data (LOD) cloud, a myriad of data elements sprung up and that revolutionized the growth of the Semantic Web both in terms of content and tools. As of now data in many domains including life science, geography, media and government became part of the LOD cloud. The proliferation of LOD cloud has been the inspiration of developing new tools and customizing the existing ones in order to effectively deal with the Semantic Web Data. Some examples of such tools are D2R server[1] , OWLIM[2] and Virtuoso[3] . In the LOD realm, a dataset is usually published by establishing links with other existing relevant datasets. This linking is the

[1] http://d2rq.org/d2r-server

[2] http://www.ontotext.com/owlim/

[3] http://virtuoso.openlinksw.com/

powerful mechanism that allows easy exploration of the interesting datasets and facts codified in them. These links can help develop applications which can take advantage of the knowledge originating from external sources.

Seismic Engineering research projects experiments generate an enormous amount of data that would benefit researchers and experimentalists working elsewhere if could be shared with their semantics. Semantics is the meaning of something, e.g., the semantics of the term experiment is a scientific research performed to conduct a controlled test or investigation. There has been an increase in the number of search on the web relevant to seismic engineering experiments and projects (Bosi et al., 2013). A couple of resources have been developed in this area to share experimental findings and outcomes, for example, Reluis[4] database. To the best of our knowledge, unfortunately, no significant effort has been devoted yet to promote access to and to integrate seismic engineering projects experimental information.

Semantic Web community has been working in order to solve data integration issue since the beginning of the last decade by employing a novel approach that incorporates the use of ontology and the Semantic Web languages, i.e., RDF and OWL. Ontology is an artifact used to model the real world facts and entities. RDF is an acronym for Resource Description Framework used to represent ontologies which do not consist of complex logical formulas. OWL, which is an acronym for Web Ontology Language, was designed to make possible the representation of comparatively complex logical formulas. Ontologies are intended to be stored in the Knowledge Base (KB), which can offer better user experience by supporting reasoning over ontological data and semantics. As KB systems can also manage the semantics of the data, they have the potential to tackle the semantic interoperability issue.

In fact, ontology is a key technique by which one can annotate semantics and provide a common, comprehensible foundation for resources on the Semantic Web. However, the development of the domain ontology requires expertise both in the domain to model as well as in the ontology development. This means that people from very different backgrounds, such as Seismic Engineering and Computer Science should be involved in the process of creating ontology.

[4] http://143.225.144.144/reluis/

Several methodologies have been developed to build ontologies (Denicola et al., 2009; Sure et al., 2003). DERA methodology (Giunchiglia and Dutta, 2011), which was developed at the University of Trento is gaining popularity because of its ease of use. As like as knowledge, ontologies also evolve as new facts can emerge at any time. This demands the continuous update of the ontology. Fulfilling this very demand is challenging in either ways, be it manual or automatic. It is hardly affordable for a research group to employ an ontology developer for a long period, though this approach would give us required accuracy. On the other hand, automatic approach is error prone. However, the latter approach is the most widely used technique in such a situation. Supervised machine learning approach can be used for keeping the knowledge updated.

The Semantic Web technologies are fostering to accept a new computing paradigm that entails a shift from databases to Knowledge Bases. There the core is the ontology that plays a main role in enabling reasoning power that can make implicit facts explicit; in order to produce better results for users. In addition, KB-based systems provide mechanisms to manage information and semantics thereof, that can make systems semantically interoperable and as such can exchange and share data between them. In order to exploit the benefits offered by state of the art technologies, we moved to KB-based system in managing data of the Seismic Engineering Research Projects and Experiments domain. To enable our system automatically exploring the relevant new datasets from the external sources, we connected the projects and experimental data to the LOD cloud.

1.2 The Problem

Employing Semantic Web tools for developing applications and Publishing data on the LOD cloud in the field of Seismic Engineering experience the following research issues.

1.2.1 Deficiency of domain ontology for categorizing information of Seismic Engineering projects and experiments: In the last couple of decades, database technologies have been advancing continuously and showing their potential in deal-

ing with large collection of data, but they were not originally designed to deal with the semantics of data. Managing data with the Semantic Web tools offers a number of advantages over Database tools in classifying, matching, mapping and querying data. While Semantic Web tools play the role of catalyst, domain specific ontologies are the key elements to perform these operations effectively. Unfortunately, it still lacks such ontology that can be used for representing information related to Seismic Engineering projects and experiments.

1.2.2 Lack of suitable vocabulary for publishing Seismic Engineering experimental data on the LOD cloud: The vocabulary is an essential component that can guide a data engineer when modeling data in RDF to publish them as part of the LOD cloud. Use of standard domain specific vocabularies is recommended as it leads to an easier consumption of the data by LOD applications and users. Despite the fact that the seismic engineering community is nontrivially contributing to the cloud, finding datasets for experiments such as dynamic tests, pseudo-dynamic tests and cyclic tests is a far cry from what has been expected. As a matter of fact, publishing such experimental data has largely been overlooked and, as such, to the best of our knowledge no vocabulary is yet developed in this field, to model data in RDF.

1.2.3 Ontology integration and linking data elements to the LOD cloud: to manage the data of a specific field of interest, domain specific ontologies provide essential support. However, they alone can hardly be sufficient to assign meaning also to the generic terms that often appear in a data source. That necessitates the use of both the generic ontology and the domain specific one. To provide seamless access to these ontologies, it is crucial to integrate them and put them in the same knowledge base. Through integration we can also avoid having duplicate concepts in the knowledge base. Because of the polysemous nature of the natural language terms finding the right correspondences between ontologies appears as a challenge. Polysemous nature of the terms in the integrated ontology pause further challenge when we try to match them with the existing datasets, e.g., DBPedia, on the LOD cloud. Usually a term with different meanings of the source matches with the same term of the target.

1.3 Solution

To address the issues described in Section 1.2, in this thesis we have proposed the development of a domain ontology that can cover the specificity of the Seismic Engineering research projects and experiments (solution to the problem 1.2.1), the specification of a vocabulary taking into account the reuse of the existing terms whenever possible (solution to the problem 1.2.2) and the application of semantic similarity measure while matching the ontological concepts and terms to the datasets of the LOD cloud (solution to the problem 1.2.3).

This thesis presents the development of a Seismic Engineering Research Projects and Experiments Ontology (SEPREMO) with a focus on research project management and experiments. The developed ontology was validated by domain experts, published in RDF and integrated into WordNet. Data originating from scientific experiments such as cyclic and pseudodynamic tests were also published in RDF. We exploited the power of Semantic Web technologies, namely Jena, Virtuoso and Virt-Graph tools in order to publish, storage and manage RDF data, respectively. Finally, a system was developed with the full integration of ontology, experimental data and tools, to evaluate the effectiveness of the KB-based approach; it yielded favorable outcomes.

Linked Open Data Cloud opened up the opportunity for researchers, experimentalists, data scientists, data practitioners and many others from government, public and private sectors for unlimited share, use and reuse of datasets. This global initiative fosters data accessibility, availability and interoperability. In a few years the LOD Cloud proliferated from some hundred datasets to a very large collection; as of March 2014, it consists of around 9k datasets covering almost all possible top level domains such as space, time, science, engineering, medicine, sports and entertainment. However, publishing Seismic Engineering research projects and experiments data has largely been overlooked and, as such, no vocabulary is yet developed in this field, to the best of our knowledge, to model data in RDF. In this thesis, we present an approach for publishing them into the LOD Cloud. In order to model the concepts complementing the vocabulary that we need for the experimental data representation, we suitably extended the SEPREMO ontology.

Moreover, we have developed a matching algorithm that takes into account the textual description of the terms and the context in which the terms are found both in the source and target datasets. In addition to these features, in measuring the similarity we also check the existence of the semantically equivalent terms.

1.4 Structure of the Thesis

The remainder of this thesis is structured as follows.

Chapter 2 discusses an Italian national project (RELUIS) database, based on which we got some concepts and entities for the SEPREMO ontology.

Chapter 3 presents an overview of ontology and semantic web, sets out the definitions, structure and some methodologies of ontology development. In addition, it also gives definitions of ontology mapping and other operations, such as ontology alignment and how it can be used. It then offers a clear description and comparison of ontology languages such as RDF, RDF(S), OWL and SKOS. Finally, we conclude this chapter with an overview of big data that is a popular term used to describe the exponential growth and availability of data, both structured and unstructured, and overview of some tools that manage big data.

In **Chapter 4**, the DERA methodology is described, which is used for building domain specific ontologies. Then, it describes the Knowledge Representation Languages RDF and OWL in terms of their capacity in representing ontologies of various kind. Afterwards, the process of integrating the developed ontology with Wordnet is explained. Basically, we applied the semi-automatic ontology integration algorithm proposed in (Farazi et al., 2011). Also the ontology matching algorithms is discussed, which will help in obtaining a high quality results. It also provides approaches to map between ontologies. Finally, evolution of methodology shows that the proposed methodology is capable of dealing sufficiently with different real word scenarios.

Chapter 5 contains the formalization of seismic engineering terminologies using the Semantic web languages. The schema is defined in such a way that it can be combined with vocabularies as produced by the developed methods. An overview of SEPREMO and the actual schema produced is provided in Appendix A.

In **Chapter 6**, overviews of the implementation of the semantic web matcher includes algorithm and system design is provided; the proposed approach elaborates how to construct dynamic semantic data linking by taking advantage of DERI pipe (**?**) features.

Chapter 7 includes reviews related to the significance of annotations in the field of information retrieval and recent research enhancements with a special focus on those that take advantage of semantic web technologies in the Seismic engineering field. The annotation and search modules of the proposed framework are implemented using Apache Lucene.

The implementation of the developed system is described in **Chapter 8**. The full implementation is presented with some case studies, to provide a full picture of the present approach and show its ability to produce high quality results. The implementation process, where the features of ontology alignments are integrated with WordNet in order to empower the search module to take advantage of the knowledge, is presented via an ontology. Besides, the RELUIS project outcome is also commented in this chapter.

Chapter 9 concludes the thesis with a summary of the work presented, its comparisons to the closest related approaches, and outlines some future directions.

CHAPTER 2

BACKGROUND STUDIES

2.1 Introduction

At present, different laboratories of Italian Universities store and manage experimental data in various fashions. Each laboratory deals with data with a unique local data model and user interface, language and scheme. Therefore, the dissemination and use of these experimental results outside the laboratory where they are produced can be problematic. To address the issue, there is an urgent need of creating a unique platform for Italian Universities Laboratories capable of sharing seismic experimental data and knowledge. Therefore, a central database where centralized access to database nodes that are distributed over the network is needed. This database will be able to connect with a central portal in a uniform manner.

The most important components of the RELUIS database given below:

- Data Access Portal. It provides a centralized access to all the projects the RELUIS laboratories make public. The Data Access Portal presents the information of the available projects, by following the structure of the Exchange Data Format. Each individual laboratory can select which projects or project results to make public.

- Exchange Data Format. This is the format in which the data and other information is stored (locally) and presented by the Data Access Portal.

- Local database. It is the local repository where data is stored.
- Web Services. Allow the exchange of content and configuration between the Data Access Portal and the local data-bases.

Section 2.2 describes the data format that is used in the communication between every RELUIS partner and the central site containing the Data Access Portal. Section 2.3 explains the RELUIS database from the perspective of external users and how they can take advantage of this RELUIS infrastructure. Section 2.4 presents distributed database architecture and Finally, Section 2.5 presents conclusions.

2.2 Exchange Data Format

The Exchange Data Format (EDF) is the format in which data are presented through the Data Access Portal (DAP) as well as the format in which data are stored locally at individual sites. The EDF has been designed to:

i. Be suitable for any experimental data type: data produced by centrifuges, reaction walls, shaking tables and so on.

ii. Allow storing data along with all other types of information (documents, image and so on.) which are useful to describe, repeat or simulate the experiments under the same conditions.

iii. Allow for data accessibility restrictions: projects can be public, restricted only to partners or, completely private (accessible only to the laboratory where have been produced).

Figure 2.1 consisting of Project, Specimen, Experiment/Computation and Signal that has then been selected for the Exchange Data Format.

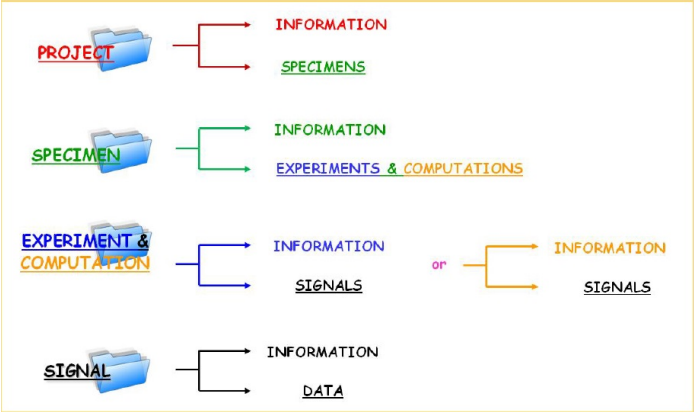


Figure 2.1: Data hierarchy

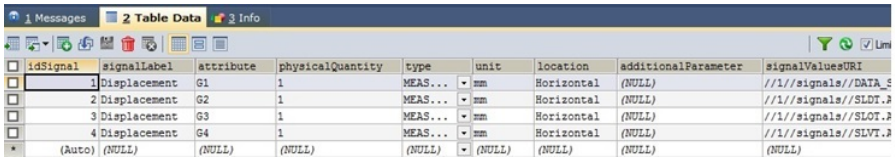
Project level includes infrastructures and persons involved and scope of the project. For the sake of uniformity, most of the fields have a fixed list of possible entries from which the user can chose. This allows for avoiding typos or using different naming for same objects, while simplifying retrieval of data and information through the search functionality. The main focus of the research project is indicated, a list of keywords to define the research areas will be provided. Moreover, it is important to have a template to fully define a report: title, author, abstract, date of publication, and link to the effective report in pdf and to the report in its original format.

A project usually includes testing of more than one physical (or numerical) structure (a short bridge pier and a tall one, several masonry structures made by different kinds of clay) identified as Specimen. It is also possible to test the same structure but in different states for example the structure in its original state and then after different types of retrofitting. While it may be argued that, in this case, all tests are performed on the same specimen, the hierarchical structure of the database demands that retrofitted specimens are included as new specimen. At this level, the physical and mechanical characteristics of the specimen are specified. Each structure is subdivided into structural elements (as for example beam, column). Nominal mechanical properties and, when experimentally measured, also actual ones can be specified. Furthermore, maximum dimensions of the specimen are specified. A comprehensive description of the geometry and dimensions is reported in the document that provide

all the necessary information for external users to adequately model the specimen; these documents show also the geometry of the facility and the location of the specimen in the facility.

In the case of a physical experiment, the same specimen is usually subjected to several types of tests that differ by the type of load imposed (quasi-static test, pseudo-dynamic test, shake table test, hammer test, etc. with or without sub-structuring, in-situ or in laboratory), by the location of the loading and/or by the configuration of the sensors. The original load time-histories and the effective inputs used on the different experiments must be explicitly identified. For example, in case of seismic experiments, the same accelerogram can be used several times by changing its intensity, or a different one may be used for each test. The original signals are preserved by providing some information on their nature (natural for accelerogram, natural-normalized for natural accelerogram normalized in the intensity, natural-modified for natural accelerogram modified according to Eurocode, etc.) and the peak excitation. A key issue is the link between experiment, sensors and signals: signals are the product of sensors during an experiment. Therefore, signals are defined by two variables: experiment and sensors.

- If a signal is issued from a direct measurement, the relationship with the sensor is obvious and should be maintained.
- If the signal results from data processing (for instance modal frequency, target displacement for a PsD algorithm, inter-story drift, etc.), the link with sensors is complex and cannot be expressed by means of a one-to-one relationship.


1 Messages | 2 Table Data | 3 Info

idSignal	signalLabel	attribute	physicalQuantity	type	unit	location	additionalParameter	signalValuesURI
1	Displacement	G1	1	MEAS...	mm	Horizontal	(NULL)	//1//signals//DATA_S
2	Displacement	G2	1	MEAS...	mm	Horizontal	(NULL)	//1//signals//SLDT.R
3	Displacement	G3	1	MEAS...	mm	Horizontal	(NULL)	//1//signals//SLOT.R
4	Displacement	G4	1	MEAS...	mm	Horizontal	(NULL)	//1//signals//SLVT.R
(Auto)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

Figure 2.2: Example of Signal Table

Each experiment has a sensors table, and a signals table which usually has more lines (i.e. signals table is given by the sum of direct measurements + processed

data). In order to validate the aforementioned database interface, the experimental data collected during the INDUSE experimental program was uploaded to the RELUIS database (see Figure 2.2). The Location is a physical descriptor of where the sensor is actually located in the specimen (e.g., first floor left bay, second floor central bay), and provides an immediate way of locating the sensor in the specimen. The coordinates of the sensor provide useful information when used in the context of a numerical model or a drawing produced by a CAD software.

The original load signal can then be scaled in intensity or applied in different direction this represents the effective input that has also to be provided. The results of all the experiments performed on a specimen are often collected in a specific specimen report.

The laboratory database located at each site adopts the very same Exchange Data Format, with the addition of some extra fields which allow the description of the characteristics and configuration of devices and sensors employed in testing. As this information is considered meaningful to (and in some cases, understandable by) only the laboratory personnel that performed the experiment, it is not made available to external users.

In the case of numerical simulation results being introduced in the database, the computer system and software used must be specified, along with detailed information on issues regarding modelling the structure (models, assumptions and so on).

At the bottom of the hierarchy is the Signal level presented in Figure 2.1. Each signal is delivered together with data regarding its units, the nature of the signal (force, acceleration), the location and the associated time sequence. In the local site database each measured signal is reported along with the associated sensor. Signals resulting from data processing or computation (for instance modal frequency, target displacement for a pseudo dynamic algorithm, etc.) are stored as computed ones.

The design of the Exchange Data Format allows for additional documentation, photos, and videos to be stored at each level.

2.3 System Architecture

The main idea in structuring the database was to store the basic data, provided by the researchers (in papers, reports, etc.), but also to be able to provide the derived data, which may assist researchers in their analyses (i.e. developing seismic performance models for different RC load bearing elements). Extracted and post-processed data may be used for various statistical studies in a user-friendly way, and for developing databases for using in a research and for developing performance/capacity models of structural elements.

2.3.1 Local site management of RELUIS Database

The standardization of the Exchange Data Format has been an iterative process involving all laboratories, especially for the part concerning the definition of a common naming which could accommodate the heterogeneity of the data encountered in the different laboratories. Once the Exchange Data Format has been defined, it was implemented in a MySQL database and tested with real experimental data. Figure 2.3 presents screenshot of the interface to the RELUIS database with laboratory data, corresponding to the Project level and the signal level. For each project, the relevant information is specified, together with the privacy restriction. MySQL Workbench or SQLyog was initially used to input information into the database, although using this generalized user interface for data manipulation appeared to be tedious and error prone, considering that just one complete experiment consists interconnected records comprising signals, sensors, configurations, materials and other metadata. Therefore, a formal process definition for the automatic conversion of laboratory data into the common format and specialized tools for its implementation have been developed, consisting of two main logical layers.

- An intermediate portable experiment format enabling the expression and storage of proprietary experimental structures in a common specification.
- Specialized interfaces and tools that allow the local users to automatically import the portable experiment files and easily manage the database.

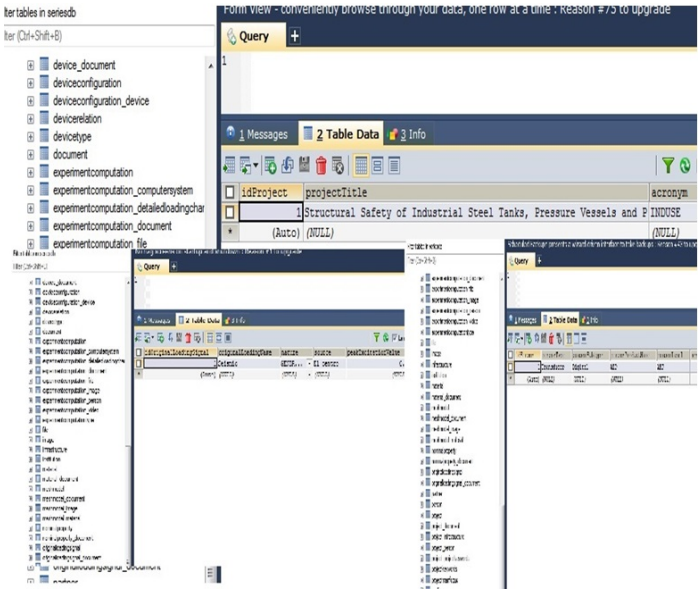


Figure 2.3: Project, Sensor level in MySQL

2.4 Distributed Database Architecture

A schematic of the distributed database in depicted in Figure 2.4, it is presented like a centralized database to external users through the Data Access Portal, it is actually a time-evolving aggregated collection of experimental data, which are regularly retrieved and updated from local distributed repositories. The aggregation of publicly shared data is performed by the Web Services installed at each local node and their communication with the Central Site.

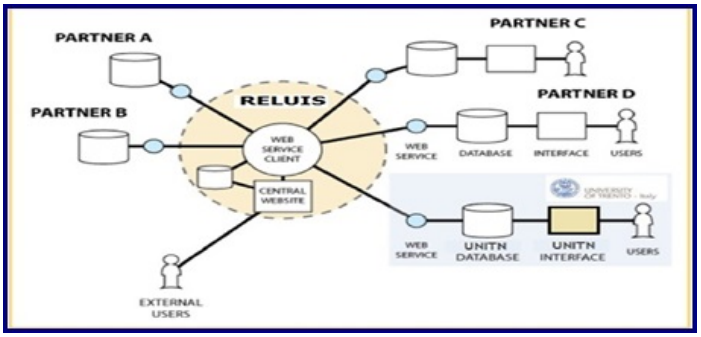


Figure 2.4: RELUIS distributed database

Data flow from the RELUIS database to the external user is given below:

- A laboratory produces experimental data and stores them in its local database. At this stage only the local users can access the data.
- The Web Services implemented at the local site automatically make available for the central site the experimental data which have been flagged as public in the local database.
- The Data Access Portal Central Site communicates regularly with individual nodes to retrieve updated information or new data.
- The information retrieved is then made publicly available in the Data Access Portal.
- External users may access, explore and finally download the published local experimental data, through the Data Access Portal.

RELUIS targeted at creating an Italian platform for wide sharing of experimental data and knowledge amongst different university, research and industry, which could be maintained and enhanced over time. The interface of the RELUIS presented in the Results chapter. This interface is designed to enable:

- Database access: functionalities to interact with the whole database internal structures in a user friendly way. Users just need to use a visually appealing interface to create, edit or delete elements in the database without knowing how the database is actually implemented. They can also conduct other tasks such as visualize data.
- Management of local users: UI allows different local users to access the database. Every user has a role assigned (administrator, contributor or guest) that enables them to use different functionalities within the interface. For instance, guest users can only visualize data, but they cannot modify any information.
- Advance tools: to extend the functionality of the system by supporting data migration, automatic input of large sets of information, visualization of signal data,

etc.

A key role is played by Web Services (WS). Within distributed systems, such as the one we find in RELUIS, SOA (Service Oriented Architecture) is an architectural paradigm that focuses in connecting heterogeneous systems under the control of different owners. This methodology allows interoperability between different systems. The table Server could also allow future services. For example, imagine a service that provides communication partner-to-partner, via the Central Site, in order to exchange information or a service that puts two or more partners in contact to configure a distributed test before conducting it (in the configuration stage, time is not critical). Basically, this service would be useful to locate other partners and authenticate them, in a centralized way.

The Web Service in the Central Site is in charge of connecting with all partners in order to get the information that feeds the Distributed Database. It translates all the received information, coming in a common agreed format to the data for the Central Database. As long as partners implement a Web Service consumer that complies with the WS specification, the platform and programming language that are employed are of no consequence. One of the benefits of Web Services is this freedom to choose. One of the typical issues about Web Services is whether it is better to create the code first or the contract first. In a typical situation, the steps involved in a Web Service creation are:

- Server creates and implements a Web service interface for an existing application.
- Server distributes a WSDL contract to use the Web Service.
- Finally, Client obtains the WSDL contract to access the Web Service.

This way of developing a Web Service is far easier than creating the WSDL directly.

Security should be conscientiously implemented on the Web Service. The Central Site implements security, each partner has the responsibility of ensuring the security

of their own Web Service. For example, most of the input data will come from the Central Website. This input, the Central Web Service might need to communicate with some partners Web Services. If the Central Site does not filter the input received from the Central Website, it can propagate a security risk to the partners Web Sites. The communication between Central Site and partners should be safe and reliable. If the Central Site just transmits user requests without checking them, neither safety nor reliability will be achieved.

2.5 Conclusion

The chapter describes the principle and associated elements which constitute RELUIS database. An Exchange Data Format that could host heterogeneous experimental data and provide all the information needed to reproduce a test, has been developed and agreed. Data stored at local sites is made accessible to external users by means of the Data Access Portal hosted at the University of Trento. In this way a centralized access to database nodes that are distributed over a network and are able to dialog with a central portal in a uniform manner, is provided. Moreover, RELUIS database enables a wider sharing of data and knowledge and ultimately, offers an unprecedented service to the earthquake engineering community. RELUIS users will be able to have access to a wide database of experimental data and information, without violating the ownership of the data that will remain with the local laboratory where data have been produced.

CHAPTER 3

STATE OF THE ART

3.1 Introduction

Starting with the history and definitions of ontology, this chapter discusses the state of the art methodologies for developing ontologies. In this chapter, we also provide a detailed description of the formalisms for representing ontologies. The discussion about ontology matching techniques is followed by the query formulation and answering in the ontology based systems. We also describe the layered architecture of the Semantic Web. Finally, we discuss data science that deals with the technologies and tools for managing large amount of data.

Section 3.2 describes what an ontology is from the perspective of Computer Science and Philosophy. Section 3.2.1 deals with the methodologies normally used to define ontologies. Section 3.2.2 focuses on ontology languages that allow the encoding of knowledge about specific domain and the main differences and similarities between the most relevant ontology languages. Section 3.2.3 presents a literature survey of ontology matching techniques. Query formulation for user query mainly presented in Section 3.2.4. Section 3.3 briefly introduces the semantic web and semantic web tools for ontology development. Section 3.4 provides a literature review challenges to management in big data science. Finally, we conclude Section 3.5 with a summary of the ontology development methodologies and semantic web tools in the current state of the art.

3.2 Ontology

Ontology can be seen as an artifact used for managing semantics of the natural language terms, which are often dubbed as concepts, and the relations between the terms in the scope of a domain. The term ontology originated from the branch of Philosophy known as Metaphysics, in which Aristotle first proposed ontology as the science in the Metaphysics(Warrington, John , 1956) that the study of being and reality for the classification of entities within a hierarchy to be capable to answer the question whether something exists. The word ontology comes from two Greek words namely: onto which means existence or being and logia which means science or study. Some essential ontological pairs are: universals and particulars, substance and accident, abstract and concrete objects, essence and existence, determinism and indeterminism. Greek Eleatic philosopher Parmendies was first introduce an ontological characterization of the fundamental type of existence where he describes two views of existence one initially and another one nothing comes from nothing. Moreover, Plato a philosopher as well as mathematician, develop a method where he distinction between true reality and illusion and he also assume that all nouns specify entities. A.N. Whitehead stated that ontology is useful to distinguish the terms ”reality” and ”actuality”. Philosophers classified ontologies in various ways for example:

- Upper ontology: Each group of ontology engineer would need to perform the task of making its terms and concepts compatible with those of other such groups only once.
- Domain ontology: Concepts relevant to a particular topic or area of interest. For example, Seismic engineering.
- Interface ontology: concepts relevant to a particular point in events of two disciplines.
- Process ontology: inputs, outputs, constraints, sequencing information involved in business or engineering process.

In the 18th century Scottish philosopher introduces Bundle theory where an object consists of only a collection properties, relation or tropes. Hence, there cannot be an object without properties nor can one even conceive of such an object. For example, a car is really a collection of the properties color, model, and capacity and so on. In particular, there is no substance in which the properties inhere. These all factors are considered in ontology development.

Dialectics is the Socratic method of reasoning which aims to understand things correctly in all movements, changes and interconnections. Its origins in ancient society, both among the Chinese and the Greeks, where thinkers sought to understand Nature as a whole, and saw that everything is fluid, constantly changing, coming into being and passing away. The key notion in dialectics is that changes occurring in a system are a result of the relationship between subsystems of the system. The correctness of dialectical reasoning is guaranteed by its ontological foundation and also deals with the categories and their sub categories into species. German philosopher Hegel identified dialectic as the tendency of a notion to pass over into its own negation as the result of conflict between its inherent contradictory forms. Afterwards, Karl Marx and Friedrich Engels adopted Hegels definition and applied it to social and economic a process that is classified as modern philosophy. Dialectic is useful to ontology development in two respects.

(i) with a view to seeing whether a claim or its contradictory is true or false.

(ii) the correctness of dialectical reasoning is guaranteed by its ontological foundation

A Conceptual metaphor in which one idea is understood in terms of another. In Metaphors We Live By (1980), George Lakoff and Mark Johnson identify three overlapping categories of conceptual metaphors:

- Orientational metaphor
- Ontological metaphor and
- Structural metaphor

In cognitive science, the conceptual domain from which we draw metaphorical expressions to understand another conceptual domain is known as the source domain. The source domain consists of a set of literal entities, attributes, processes and relationships, linked semantically and apparently stored together in the mind. The conceptual domain that is understood in this way is the target domain. Thus the source domain is commonly used to explain the target domain. To know a conceptual metaphor is to know the set of mappings that applies to a given source-target pairing. For example, the theory was not intended to account for language in use. Conceptual metaphor also helps to generalize the concept for example polysemy generalization, semantic change and inferential generalization. Moreover, metaphoric concepts are expressed through terms that express explicitly the two concepts that play a part in a metaphor, and are represented in unique formats. The conceptual metaphor approach is for identifying underlying meaning of concept of the given domain.

Moreover, the Values Theory defines values as desirable, trans-situational goals, varying in importance, which serves as guiding principles in ontology development. The crucial content aspect that distinguishes among values is the type of motivational goal they express. In general, values theory differentiates between moral and natural concepts. For example the statement John is good person represents a very different sense of the word good than the statement That was some good food.

Whereas during the 1990s, this word became relevant for the knowledge engineering community.

Recently, ontology became a popular research topic in many areas, including e-commerce (Hepp , 2008), knowledge management (Davies and Weeks , 2004), earthquake engineering (Hasan et al., 2013), and natural language processing (Fensel , 2001). In this context, ontology is an explicit specification of a conceptualization (Gruber , 1993); this implies that the modeling provided by ontology should specify a systematic correlation between reality and its representation. Conceptualization is an abstract, simplified view of the world that present for some purpose. Ontologies aim at overcoming the problem of implicit and hidden knowledge by making the conceptualization of a domain explicit. It is also used to make assumptions about the meaning of a specific concept. It can also be seen as an explication of the context for

which the concept is normally used. Moreover, everything (i.e., any knowledge-based system or any knowledge-level agent) is liable to some conceptualization, explicitly or implicitly. Therefore, since there is consensus of terms, it is a shared conceptualization. More formally, an ontology defines the vocabulary of a problem domain and a set of constraints (axioms or rules) on how terms can be combined to model specific domains. It is typically structured as a set of concept definitions and relations between them. Hence, Ontologies are machine process able models that provide the semantic context, enabling natural language processing, reasoning capabilities, domain enrichment and domain validation.

Guarino and Giaretta (Guarino and Giaretta, 1995) collected the following seven definitions:

- Ontology as a Philosophical discipline
- Ontology as an informal conceptual system
- Ontology as a formal semantic account
- Ontology as specification of a conceptualization
- Ontology as representation of a conceptual system via logical theory
- Characterized by specific formal properties
- Characterized only by its specific purpose
- Ontology as the vocabulary use by a logical theory
- Ontology as specification of a logical theory

The invention of the Semantic Web provide a set of standards where ontologies are the principal resource to integrate and deal with information. Over the past years, many representation languages have been developed for ontologies, some of which are highly efficient, standardized, and relevant to the present research are in fact the Resource Description Framework (RDF), andthe most recent Web Ontology Language (OWL). Furthermore, they enable the separation of domain knowledge from

operational knowledge and the reuse of domain and operational knowledge separately (e.g., configuration based on constraints), and can manage combinatorial explosion and enable automated reasoning.

The purpose of an ontology is not to model the whole world, but rather a part of domain. A domain is just a specific subject area or area of knowledge, like medicine, earthquake engineering, realestate,geo names, financial management and so on.

3.2.1 Ontology Design and development

Ontology building is a complex process and challenging task. Furthermore there are no standard methodologies for building ontology therefore, finding an adequate methodology was not easy. To address this point, Gruber has listed a number of principles for the design of ontologies such as clarity, coherence, extensibility, minimal encoding bias and minimal ontological commitment (Gruber , 1993).The development of domain ontology is known as Ontological Engineering, which is a continuous process incorporating the complete life-cycle of an ontology; an ontological engineering process typically comprises activities such as: Purpose Identification and Requirements Specification, Knowledge acquisition, Conceptualization, Reuse and Integration, Evaluation and Documentation (Falbo et al. , 2002; Perez et al. , 2004). Each support activity is carried out during a specific part of the complete development process, but they are all essential to the development process. In the following subsections, we will present several types of ontology engineering methodologies.

METHONTOLOGY

The METHONTOLOGY methodology is presented by (Fernandez et al. , 1994).It is one of the earlier attempts to develop a method specifically for ontology engineering processes (prior methods often include ontology engineering as a sub-discipline within knowledge management). An ontology lifecycle consisting of a number of following sequential work phases or stages:

(i)**Specification**: Identify purpose, scope and granularities. This phase is essential for design, evaluation and reuse of ontologies.

(ii)**Knowledge Acquisition** : Once the domain or scope of an ontology has been decided, the process of acquiring domain knowledge from specialists (in our domain earthquake engineer and mechanical engineer); database metadata; standard text books; research papers and other ontologies.

(iii)**Conceptualization**: The main activities in conceptualization are:

- identification of concepts and their properties
- classification of groups of concepts in classification trees
- description of properties
- identification of instances
- description of instances.

(iv)**Integration**: Use or combine available data from existing ontologies for example WordNet, DBpedia to obtain a consistent ontology.

(v)**Evaluation**: By assessing the competency of the ontology to satisfy the requirements of its application, including determining the consistency, completeness and conciseness of an ontology (Perez , 1994). We evaluate ontologies for completeness, consistence and avoidance of redundancy

(vi)**Documentation**: An ontology that cannot be understood cannot be reused. Informal and formal complete definitions, assumptions and examples are essential to promote the appropriate use and reuse of ontology.

On-To-Knowledge

The On-To-Knowledge Methodology (OTKM) (Sure et al. , 2003) is, similarly to METHONTOLOGY, a methodology for ontology engineering that covers the big steps, but leaves out the detailed specifics. OTKM is framed as covering both ontology engineering and a larger perspective on knowledge management and knowledge processes, but it heavily emphasizes the ontology development activities and tasks. The method prescribes a set of sequential phases: Kickoff, Refinement, Evaluation, and

Application and Evolution.

DERA

To gain satisfactory result for ontology development we found DERA methodology. This methodology allows for building domain specific ontologies. Domain is an area of knowledge in which users are interested in. For example, earthquake engineering, oceanography, mathematics and computer science can be considered as domains. In DERA, a domain is represented as a 3-tuple D = $<$E, R, A $>$, where E is a set of entity-classes that consists of concepts and entities; R is a set of relations that can be held between concepts and entities and A is a set of attributes of the entities. Moreover, DERA accepts fully automated reasoning by direct encoding in Description Logics (DL) (Baader et al. , 2003).

In this three basic components concepts, relations and attributes are organized into facets; hence, the ontology is based on faceted methodology. Facet is a hierarchy of homogeneous concepts describing an aspect of a domain. S. R. Ranganathan, who was an Indian mathematician-librarian, was the first to introduced faceted approach capable of categorizing books in the libraries (Ranganathan , 1967).

The mapping above 3-tuple to DL should be obvious. IS-A, part-of and value-of relations form the backbone of facets, are assumed to be transitive and asymmetric, and hence are said to be hierarchical. Other relations, defined, not having such properties are said to be associative and connect terms in different facets. All together facets constitute the TBox of a descriptive ontology. The main steps in the methodology are as follows:

- Identification of the atomic concepts
- Analysis
- Synthesis
- Standardization
- Ordering

- Formalization

During the early stage of ontology development research, Gruber provides five design principles (Gruber , 1993):

- **Clarity**: communicate effectively the intended meaning of defined terms. Definitions should be objective, complete and documented with natural language.
- **Coherence**: inferences that are consistent with the definitions. If a sentence inferred from the axioms contradicts a definition then the ontology is incoherent.
- **Extendibility**: enable the definition of new terms for special uses based on the existing vocabulary and that avoids the revision of the existing vocabulary.
- **Minimal encoding bias**: Specified at the knowledge level without depending on a particular symbol level encoding.
- **Minimal ontological commitment**: specify the weakest theory and define only those terms those are essential to the communication of knowledge consistent with the theory.

In this thesis the focus is mainly on the development activities; providing semi-automatic support for some of the activities during development. Several of the support activities are also highly relevant, such as knowledge acquisition, integration, and evaluation. To conclude, we use the **DERA** methodology for our ontology development.

3.2.2 Ontology Representation

The ontology must be specified and encoded, that is, delivered using some concrete representation. There are a variety of languages which can be used for representation of conceptual models, with varying characteristics in terms of their expressiveness, ease of use and computational complexity. In this section more information

on types of ontology representation such as RDF, RDFS, OWL and SKOS is presented.

3.2.2.1 RDF

The Resource Description Framework (RDF) is a data model used to represent information about resources in the World Wide Web (WWW) and can be used to describe the relationships between concepts and entities. It is a framework to describe metadata on the web. Three types of things are in RDF: resources (entities or concepts) that exist in the real world, global names for resources (i.e. URIs) that identify entire web sites as well as web pages, and RDF statements (triples, or rows in a table) (Klyne and Carroll , 2004). Each triple includes a subject, an object and a predicate(see Figure 3.1). RDF is designed to represent knowledge in a distributed way particularly concerned with meaning.

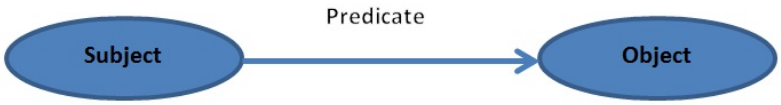


Figure 3.1: RDF Triple

From this basic structure, schemas can be built, placed on top of the RDF structure and used to build complex ontologies to help in the structuring and organization of data. Moreover, text form of RDF is called RDF serialization. It can have more forms. Among these forms is RDF/XML, N3 notation, N-triples, RDFa. Serialization called RDF/XML is the mostly used type of serialization. It is based on the XML language.

RDF can be used in several applications, one of the most important being resource discovery, used to enhance search engine capabilities. It is also used to facilitate knowledge sharing and exchange in intelligent software agents to describe the content and content relationships available with any resource, such as a page.

3.2.2.2 RDFS

RDF schema is a semantic extension of RDF which provides mechanisms to describe groups of related resources and the relationships between these resources in a RDF document (Miller and Brickley , 2002). To define the semantics of resource, RDF schema utilize superclass, class and subclass concepts which are very similar to the concept used in object oriented programming like Java. Particularly, a class contains a set of resources. Relation between classes a domain specific hierarchy is formed; the resulting hierarchy is able to restrict the interpretation of the resources to their intended semantics in a RDF document. To ensure consistency of semantic interpretation, RDF schema allows property to define its RDF and RDF schema are only capable of representing semantics.

3.2.2.3 OWL

Web Ontology Language is designed to represent comparatively complex ontological relationships and to overcome some of the limitations of RDF such as representation of specific cardinality values and disjointness relationship between classes (Giunchiglia et al. , 2010). The language is characterized by formal semantics and RDF/XML based serializations for the web. As an ontology representation language, OWL is essentially concerned with defining terms that can be used in RDF documents, i.e., classes, properties and instances. It serves two purposes: first, it identified current document as an ontology and second it serves as a container metadata regarding the ontology. This language focuses on reasoning techniques, formal foundations and language extensions. OWL uses URI references as names and constructs these URI references in the same manner as that used by RDF. The W3C allows OWL specification includes the definition of three variants of OWL, with different levels of expressiveness. These are OWL Lite, OWL DL and OWL Full ordered by increasing expressiveness.

3.2.2.4 SKOS

Simple Knowledge Organization System (SKOS) is a model for expressing knowledge organization systems in a machine-understandable way, within the framework of the Semantic Web. The SKOS Core vocabulary is an RDF application. Using RDF allows data to be linked and merged with other RDF data by Semantic Web applications. SKOS Core provides a model for expressing the basic structure and content of concept schemes, including thesauri, classification schemes, subject heading lists, taxonomies, terminologies, and other types of controlled vocabulary used for representing semantic Knowledge Organization Systems. It's being widely used beyond the librarian's world, partly because of its better labelling features (prefLabel, altLabel) that can be used with any kind of real-world data.

3.2.3 Ontology Matching Techniques

Information and communication systems are facing unprecedented levels of distribution and heterogeneity due to the advent of new technological and socio-organizational paradigms. Hence, many applications/scenarios see the ontology matching process as an appropriate approach to overcome such heterogeneity since it is able to define an alignment between two ontologies at the conceptual level, which support to enhance interoperability between applications and/or systems.

Ontology matching has been defined as finding correspondences between semantically related entities of different ontologies (Euzenat and Shvaiko , 2007). These correspondences are called alignments, and represent not only equivalence, but also other kinds of relations, such as sub-sumption, or disjointness. Ontology Matching is seen as the process of semi automatically the correspondences between semantically related ontological entities of the ontologies adopted by the organizations wishing to interoperate.

Precisely, as stated (Euzenat and Shvaiko , 2007), the matching operation determines as a function f which, from a pair of ontologies to match O1 and O2, a set of parameters p, a set of resources res and an input alignment A, it returns an alignment

A" between the matched ontologies.

$$A" = f(O1, O2, p, res, A)$$

There are some other parameters that can extend the definition of matching:

- the use of an input alignment A, which is to be extended;
- the matching parameters, for instance, weights, or thresholds; and
- external resources, such as common knowledge and domain specific thesauri

An alignment is a set of correspondences between entities belonging to the matched ontologies. Alignments 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). Moreover, alignment also expressed as a set of relations that is used to represent the relation holding between the entities (e.g. equivalence, subsumption, disjoint).

In order to align entities from ontologies in different description languages (e.g. OWL, RDF) or in the same language; alignment technique use all the features of ontologies (concept, attributes, relations, structure, etc.) to get efficiency and high quality results. For this purpose, several matching techniques have been used such as string, structure, heuristic and linguistic matching techniques with thesaurus support, as well as human intervention in certain cases, to obtain high quality results. This technique integrates some important features in matching in order to achieve high quality results, which will help when searching and exchanging information between ontologies. Moreover, an ontology alignment system illustrates the solving of the key issues related to heterogeneous ontologies, which uses combination-matching strategies to execute the ontology-matching task. Therefore, it can be used to discover the matching between ontologies.

Matchers can be classified based on many independent classifications. From the definition of the matching process introduced, the algorithms could be classified according to three relevant dimensions.

1. **Pre-processing**

The first step entails obtaining useful information from the ontologies that are to be matched, beginning by loading two ontologies and extracting useful ontological features such as class names and properties. In that respect, algorithms may support the relational, object-oriented and entity-relationship models e.g. Artemis (Castano et al. , 2000), XML and relational models (e.g. Cupid (Madhavan et al. , 2001)) or RDF and OWL models for example NOM (Ehrig and Sure, 2005), FOAM (Ehrig and Sure, 2005), FALCON-AO (Jian et al. 2005), OLA (Euzenat , 2004), oMap (Straccia and Troncy , 2005).

2. **Process Dimensions**

In general, the similarity between entities needs to be calculated in order to find the correspondence between ontology entities. For that reason, different strategies used (e.g. string similarity, synonyms, structural similarity and similarity based on instances) for achieving similarity between entities.

The first context concerns the granularity and the way algorithms interpret the input. In terms of granularity, algorithms are classified as (i) Element-level, which are those that compute correspondences by analyzing each entity individually, ignoring the existing relationships with other entities and (ii) Structure-level, which are those that compute correspondences by analyzing how entities appear together in a structure, through existing relationships between entities. With respect to the way algorithms interpret the input data, they are classified as:

- Syntactic, which are those that interpret the input regarding its sole structure through some clearly defined method;

- External, which are those that interpret the input in the light of some external resources of a domain or of common knowledge;

- Semantic, which are those that interpret the input using some formal semantics. In this case, the outputs are also justified based on the adopted formal semantics.

The second perspective is based on the type of data used as input. At a first level, it is distinguished by algorithms working on:

(i) Terminological data (i.e. strings). Terminological matchers can be classified further either as string-based (those that consider strings as sequences of characters) or as linguistic (those that consider strings as terms of natural language);

(ii) Structure (structural). The structural matchers can be classified either as internal (those that consider the internal structure such as attributes and the data types) or as relational (or external, when considering the relations an entity has with the other entities);

(iii) Models (or semantics). These matchers require a semantic interpretation of the ontologies;

(iv) Extensional (data instances). These matchers exploit the current population of the ontologies.

Basic algorithms can be multiple classified as graphically depicted in Figure 3.2 OLA (Euzenat and Shvaiko , 2007), where the first layer represents the first perspective (Granularity/Input Interpretation), the second layer represents the basic algorithms or process level and the third layer represents the second perspective (kind of input).

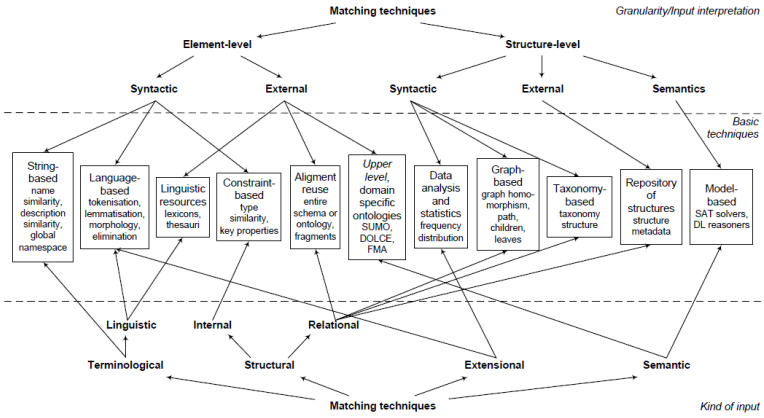


Figure 3.2: Matching algorithm

3. **Post Processing** Finally, the post processing from all matching steps is a set of alignment entities, which will be aggregated by efficient algorithms to check the correctness of alignment entity relationships and avoid redundancy.

3.2.4 Query Formulation and Answering

The main aim of the user query formulation is to have a representative and significant sample of queries reflecting users interests and needs focused on our representation of the target domain onto. The answers of these queries are then returned from the underlying data sources by taking into account the matching correspondences between domain ontology, and mappings between the ontologies and the actual data sources on the other side. In the background, queries are translated into formal languages (e.g., SQL, XQuery, or SPARQL).

SPARQL query language for matching against RDF graphs, with a syntax resembling to SQL, but which is more powerful, enabling queries spanning multiple disparate (local or remote) data sources containing heterogeneous semi-structured data. It allows for getting values from structured and semi-structured data, exploring data by querying unknown relationships, performing complex joins of disparate databases into a single one, and transforming RDF data from one vocabulary to another (Hitzler et al. , 2009). SPARQL provides definitions for:

- Simple matching of RDF data,
- The ability to combine multiple matches together,
- Matching data types such as integers, literals, etc. based on conditions such as greater than, equal to and more on.
- Optionally matching data that is, if certain data does exist it must meet a certain criteria but the query does not fail if the data doesnt exist,
- Combining RDF data sets together to query at the same time, and
- Ordering and limiting matched data.

To visualize queries several semantic web approaches for example ISPARQL [1], RDFAuthor (Miller and Brickley , 2002), GRQL (Athanasis et al. , 2004) and Nitelight (Russell et al. , 2008) propose to formulate a SPARQL query in triple patterns. Although these approaches vary in their intuitiveness they all intend to assist developers rather than end-users, as they require technical knowledge about the queried sources.

Another one, Mashup editor for example Yahoo Pipes [2] allow people to write query inside a module and visualize these modules and their inputs and outputs as boxes connected with lines. Recent approach in the semantic web community Deri Pipes[3] inspired by Yahoo's Pipes, is an engine and graphical environment for general Web Data transformations and Mashup supports RDF, XML, Microformats, JSON and binary streams. Use it as a "Web Pipe" or embedded in the applications Works as a mashup command Line tool supports SPARQL, XQUERY, Several scripting languages. Extend it as needed DERI Pipes, in general, produce as an output streams of data (e.g. XML, RDF, JSON) that can be used by applications. However, when invoked by a normal browser, they provide an end user GUI for the user to enter parameter values and browse the results.

3.3 The Semantic Web

The inventor of the Web, Tim Berners-Lee, envisioned a more organized, well connected and well integrated form of its data that are suitable for humans to read and for machines to understand (T. Berners-Lee, 1999). This new form of the Web is called the Semantic Web. With the invention of the Semantic Web, computing paradigm is experiencing a shift from databases to Knowledge Bases (KB), where ontologies play a major role in enabling inferencing that can make hidden facts unconcealed to produce better results for users.

The traditional knowledge representation methods are not applicable to the web data in an out-of-the-box manner. In such a context, Semantic web provides a com-

[1] http://lod.openlinksw.com/isparql/

[2] http://pipes.yahoo.com/pipes

[3] http://pipes.deri.org/

mon framework that allows data to be shared and reused across applications, enterprise, and community boundaries. The Semantic Web, consisting of machine processable information, will be enabled by further levels of interoperability. Figure 3.3 illustrates the architecture of the semantic web.

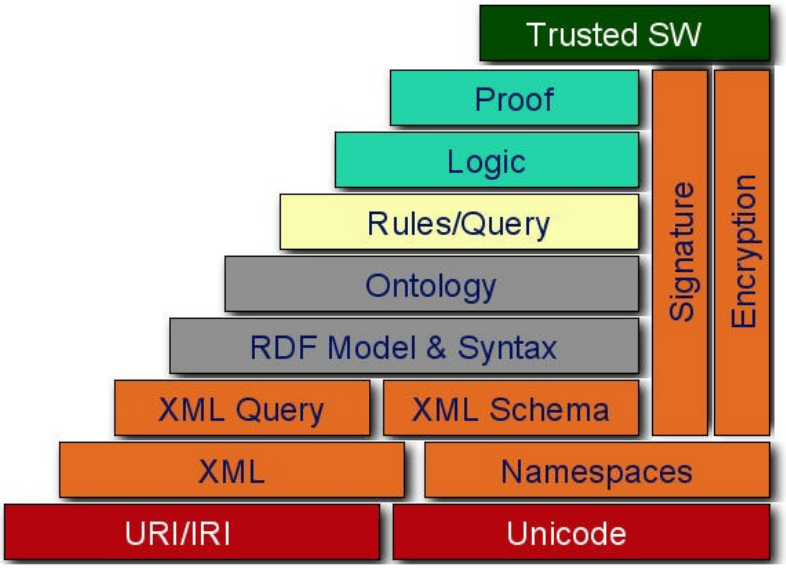


Figure 3.3: Semantic Web Architecture (Berners-Lee et al., 2001)

Some languages also known as Semantic Web languages are used to represent information about resources on the Web. This information is not limited to Web resource description, but can be about anything that can be identified. Uniform Resource Identifiers (URIs) are used to uniquely identify entities. For example, it is possible to assign a URI to a person, to the company person works for, to the experiment he/she accomplished. Therefore relations between these entities can be written and shared on the Semantic Web in unambiguous way. A stack of languages has been published as W3C recommendations to be used on the Semantic Web. We summarize these languages and their goals in the Ontology representation sections.

In the Semantic Web, the building of systems follows a logic which considers the structure of ontology. A reasoner could be used to check and resolve consistency problems and the redundancy of the concept translation. A reasoning system is used to make new inferences. Finally, concerns the trustworthiness of the information on

the web in order to provide an assurance of its quality.

Many challenging features the Semantic Web applications have to tackle in order to become truly applicable have also been addressed recently. This includes changing knowledge (Heflin and Hendler , 2000), inconsistencies (Haase et al. , 2005) or uncertainty (Bobillo and Straccia , 2008) or from the probabilistic (Peng et al. , 2005) perspective. Most approaches handling these features seek for a solution that is compatible with or an extension of the core Semantic Web standards (mainly RDF and OWL).

Most recent practice ,linked data that denotes a set of best practices for publishing data on the Semantic Web, then also called Web of Data. Moreover, linked data are usually published using vocabularies with a semantics, which enables scalable reasoning across datasets. A lot of providers have already published their data according to these principles and interlinked them with other datasets. The hub in this big picture is DBpedia[4] , a huge collection of general-purpose data extracted from a huge collection of general-purpose data extracted from the web 2.0 encyclopedia Wikipedia [5] and made available as RDF. Data from specific domains, such as scientific publications (green), biomedicine (pink), social networks (orange), multimedia (dark blue), geodata (GeoWorNet) and government statistics have also been published as linked open data. Note that linked data do not have to be open, but making datasets open of course helps to interlink and reuse knowledge; therefore, the open datasets have so far been the most visible and most widely used instances of linked data.

To support the vision of the Semantic Web which is making machine-readable content available on the Web, several software platforms and application interfaces (APIs) have been developed to permit the automatic creation and use of RDF(S) and OWL ontologies. A more exhaustive list of these platforms could be found in ((**?**); they include Protege, WebODE, OntoEdit, KAON1, and so forth. Beside the software platforms used for the edition of RDF(S) and OWL ontologies, there exist APIs such as Jena API, Sesame(Watson , 2008), Virtuoso, etc., which provide facilities for the persistence storage and query of RDF(S) and OWL ontologies. Protege and Jena API

[4]http://dbpedia.org/

[5]https://www.wikipedia.org/

are discussed in this study as they are the leading platforms for Semantic web development (Wilkinson et al. , 2003); furthermore, they are both open source software and might facilitate the repeatability of this study.

a. Protege

Protege is an open-source platform developed at Stanford Medical Informatics. It provides an internal structure called model (Knublauch et al. , 2004) for ontologies representation and an interface for the display and manipulation of the underlying model. The Protege model is used to represent ontology elements as classes, properties or slots, property characteristics such as facets and constraints, and instances. The Protege graphical user interface can be used to create classes and instances, and set class properties and restrictions on property facets. Additionally, Protege has a library of various tabs for the access, graphical visualization, and query of ontologies. Protege can be currently used to load, edit and save ontologies in different formats including XML, RDF, UML, and OWL.

b. Jena API

Jena is a Java ontology API. It provides object classes for creating and manipulating RDF graphs called interfaces. A RDF graph is called a model and represented with the Model interface. The resources, properties and literals describing RDF statements are represented with the Resource, Property and Literal interfaces respectively. Jena also provides methods that allow saving and retrieving RDF graphs to and from files. The Jena platform supports various database management systems such as PostgreSQL, MySQL, Oracle, and so on; it also provides various tools including RDQL query language, a parser for RDF/XML, I/O modules for RDF/XML output, etc. (Wilkinson et al. , 2003). To develop Earthquake engineering Research projects and experiments we used JENA API.

3.4 Data Science

Data is being generated, collected and archived in digital form in high volumes by many research groups, organizations and agencies worldwide; it can be difficult to find what you want and correctly process it to get what you need. This data can

be used to improve the experience of our lives through analysis of our consumption, interactions and behaviors; in research today, data has become a competitive advantage and necessary component of product development. Furthermore, the fast evolution of technologies/processes and the discovery of new scientific knowledge require flexibility in handling dynamic data and models in data management systems. Among others, there are three core challenges for effective data management in scientific research.

- The ability to provide a data management service that can manage large quantities of heterogeneous data in multiple formats (text, image, and video) and not be constrained to a finite set of experimental, imaging and measurement plat forms or data formats.
- The ability to support metadata-related services to provide context and structure for data within the data management service to facilitate effective search, query and dissemination
- The ability to accommodate evolving and emerging knowledge, technologies for example R[6] and Matlab[7]

3.4.1 R Statistical Tools

R is an open source statistical programming language and environment, created by Ross Ihaka and Robert Gentleman (Ihaka and Gentleman , 1996) at the University of Auckland and, since 1997, developed and maintained by the R-core group. Originally utilized in an academic environment for statistical analysis, it is now widely used in public and private sector in a broad range of fields, including informatics. The success of R can be attributed to several features including flexibility, a substantial collection of good statistical algorithms and high-quality numerical routines, the ability to easily model and handle data, numerous documentation, cross-platform compatibility, a well-designed extension system and excellent visualization capabilities to list some of the more obvious ones (Gentleman , 2008). Moreover, the application and server

[6] http://www.r-project.org/
[7] http://it.mathworks.com/

bridges the front-end Web user interface with R on the server-side in order to compare statistical macro data, and stores analyses results in RDF for future research. As a result, distributed linked statistics with accompanying data can be more easily explored and analyzed by interested parties. Earthquake engineering community has a specific focus on numerical and experimental analysis and represents a repository for hundreds of high-throughput experimental data. The development and distribution of new packages is a very dynamic and important aspect of the R software itself.

3.4.2 MatLab

Matlab is amazing tool for statistical analysis and visualization, with mature implementations for many machine learning algorithms. However, this tool is a common analysis tool used for data manipulation, signal processing and function integration. In most cases, need to mix-in various other software components in like Java or Python and integrate with data platforms like Hadoop, when building end-to-end data products.

Moreover this tool widely used data analysis, with the capability of directly handling the underlying semantic objects and their meanings. Such capabilities allow users to flexibly assign essential interaction capabilities, such as brushing-and-linking and details-on-demand interactions, to visualizations. To demonstrate the capabilities, two usage scenarios in document and graph analysis domains are presented.

3.4.3 Hadoop

The size of data sets being collected and analyzed in the industry for business intelligence, earthquake engineering research organization are growing rapidly, making traditional warehousing solutions prohibitively expensive. Hadoop is a popular open source map-reduce implementation which is being used in companies like Yahoo, Facebook etc. to store and process extremely large data sets on hardware. Hadoop was initially inspired by papers published by Google in outlining its approach to handling large amount of data, and has since become the de facto standard for storing, processing and analyzing hundreds of terabytes, and even petabytes of data. Apache Hadoop is open source and pioneered a fundamentally new way of stor-

ing and processing data. Instead of relying on expensive, proprietary hardware and different systems to store and process data, Hadoop enables distributed parallel processing of huge amounts of data across inexpensive, industry-standard servers that both store and process the data, and can scale without limits. With Hadoop, no data is too big. Hadoop has a general-purpose file system abstraction (i.e., can integrate with several storage systems such as the local file system, HDFS, Amazon S3, etc.). Hadoop family include following components:

MapReduce	Distributed computation framework
HDFS	Distributed file system
HBase	Distributed, column-oriented database
Hive	Distributed data warehouse
Pig	Higher-level data flow language and parallel execution framework
ZooKeeper	Distributed coordination service
Avro	Data serialization system (Remote procedure call (RPC) and persistent data storage)
Sqoop	Tool for bulk data transfer between structured data stores (e.g., RDBMS) and HDFS
Oozie	Complex job workflow service
Chukwa	System for collecting management data
Mahout	Machine learning and data mining library
BigTop	Packaging and testing

Table 3.1: The Hadoop Family

Main design principles for the Hadoop Eco System given bellow:

- Linear scalability
  (i) More nodes can do more work within the same time
  (ii) Linear on data size, linear on compute resources
- Move computation to data
  (i) Minimize expensive data transfers
  (ii)Data is large, programs are small
- Reliability and Availability: Hadoop is schema-less, and can absorb any type of data, structured or not, from any number of sources. Data from multiple sources can be joined and aggregated in arbitrary ways enabling deeper analyses than any one system can provide.

- Simple computational model (MapReduce)
  (i) Hides complexity in efficient execution framework
- Streaming data access (avoid random reads)
  (i) More efficient than seek-based data access

Moreover, Hadoop structures data in to the well understood database concepts like tables, columns, rows, and partitions. It supports all the major primitive types integers, floats, doubles and strings as well as complex types such as maps, lists and structs (Thusoo et al. , 2010). The query language of the Hadoop is very similar to SQL and therefore can be easily understood by anyone familiar with SQL.

Challenge in Hadoop, MapReduce is not a good match for all problems. Its good for simple requests for information and problems that can be broken up into independent units. But it is inefficient for iterative and interactive analytic tasks. MapReduce is file-intensive. Because the nodes dont intercommunicate except through sorts and shuffles, iterative algorithms require multiple map-shuffle/sort-reduce phases to complete. Another challenge the fragmented data security issues in Hadoop, though new tools and technologies are surfacing.

3.4.4 Open Refine

OpenRefine[8] (formerly Google Refine) is a powerful tool for working with messy data: cleaning it; transforming it from one format into another; extending it with web services; and linking it to databases like Freebase. OpenRefine will interest librarians, scientists, data curators, researchers, business analysts, data journalists, and digital repository managers in a variety of disciplines who need clean, usable data. OpenRefine is very powerful; Users can explore data to see the big picture, clean and transform data, and reconcile data with various web services. OpenRefine features are:

- OpenRefine works with local files or data from web addresses in a number of

[8] http://openrefine.org/

file formats, including CSV, TSV, XLS, XML, and other formats.

- It has the ability to filter or search for certain elements that need to be changed in some way, which restricts the view to just the relevant cells, rows, or columns that contain the elements. Then the user can perform the desired action on just those data.

- It can find duplicate entries, empty cells, entry variations, inconsistencies, and patterns of errors for bulk fixing and cleaning.

- It provides a quick analysis of the data contained in the file; for instance, the word facet tool can analyze the words in a column and return a count of each of the unique words, and the results sort alphabetically by default, but when sorted by count, any trends can be seen at a glance

When dealing with data, the ability to modify and transform many records at once allows users to save tremendous amounts of time and create usable data; OpenRefine tools for the data can be viewed, filtered, and modified.

3.4.5 Apache Spark

Apache Spark[9] is an open source cluster computing system that aims to make data analytics fast both run and write. Originally developed as a research project at UC Berkeley's AMPLab, the project achieved incubator status in Apache in June 2013. To run programs faster, Spark offers a general execution model that can optimize arbitrary operator graphs, and supports in-memory computing, which lets it query data faster than disk-based engines like Hadoop (Zaharia et al. , 2010).

Spark seeks to address the critical challenges for advanced analytics in Hadoop. First, Spark is designed to support in-memory processing, so developers can write iterative algorithms without writing out a result set after each pass through the data. This enables true high performance advanced analytics; for techniques like logistic regression, project sponsors report runtimes in Spark 100 times faster than what

[9] https://spark.apache.org/

they are able to achieve with MapReduce. Second, Spark offers an integrated framework for advanced analytics, including a machine learning library (MLLib); a graph engine (GraphX); a streaming analytics engine (Spark Streaming) and a fast interactive query tool (Shark). This eliminates the need to support multiple point solutions, such as Giraph, GraphLab and Tez for graph engines; Storm and S3 for streaming; or Hive and Impala for interactive queries. A single platform simplifies integration, and ensures that users can produce consistent results across different types of analysis.

At Spark's core is an abstraction layer called Resilient Distributed Datasets (RDDs). RDDs are read-only partitioned collections of records created through deterministic operations on stable data or other RDDs. RDDs include information about data lineage together with instructions for data transformation and (optional) instructions for persistence. They are designed to be fault tolerant, so that if an operation fails it can be reconstructed.

For data sources, Spark works with any file stored in HDFS, or any other storage system supported by Hadoop (including local file systems, Amazon S3, Hypertable and HBase). Hadoop supports text files, SequenceFiles and any other Hadoop InputFormat. Spark supports programming interfaces for Scala, Java, Python and R.

3.5 Conclusion

The design of ontology is to achieve a common and shared knowledge that can be disseminated between people and application systems. Furthermore, ontologies play a key role in achieving interoperability across the organization for the reason that aim to capture domain knowledge and their role is to create semantics explicitly in a generic way, providing the basis for agreement within a domain. Now a day, ontologies have become a popular research topic in many research communities. In fact, ontology is a main component of my research; therefore, the definition, structure and the main operations and applications of ontology are provided.

Ontology language is the ground of ontological knowledge systems, the definition of a system of knowledge representation language specification; it not only has a rich and intuitive ability to express and use it, but the body should be easily understood

by the computer, processing and applications. Thus, a brief survey of state-of-the-art ontology representation language which is used to express ontology over the web is provided; all relevant terms were shown in order to provide a basic understanding of ontologies which are the basis of ontology languages. Moreover, we briefly described method, techniques and frameworks for aligning ontologies. Finally, the novel statistical tools (e.g., R,Matlab, Hadoop, OpenRefine and Apache Spark) to analyze and visualize data, becoming increasingly popular tools as the data gets bigger and more distributed.

CHAPTER 4

ONTOLOGY DEVELOPMENT

4.1 Introduction

Modern information systems is moving from data-processing towards concept-processing, meaning that the basic unit of processing is less and becoming more a semantic concept which carries an interpretation and exists in a context with other concepts. Ontologies play a key role representing concept for a particular domain. Developing ontologies involves taking a domain knowledge, formalizing this knowledge into a machine computable format and encoding it in an ontology language.

Ontology building is a very complicated activity for several reasons. First, because it requires time consuming work of experts. Moreover the classification task is not simple as it seems. Finally it is complicated because of the incredible speed in which the knowledge develops itself in the real world, and the constraints that ontology engineers faces to continuously update and enrich the generated ontologies with new concepts, terms and lexicons. In this way an ontology often becomes an endless opportunity for the future development which requires constant manual efforts and resources to be built and maintained. In recent years, methods (e.g. DERA) methods have been developed to solve the problems related to manual ontology building with automatic or semi-automatic methods. The research question of this work is the following: Is it possible to substitute (fully or partially) human activity in a complex task like ontology building with an actual method? We will try to explain this question through experimental result conducted on a concrete example where a manual domain specific ontology has been compared with a semi-automatically built one.

The goal of this work is to present a concrete example regarding the evaluation of the semiautomatic approach to ontology building compared with the manual one. This thesis work has been developed on a three phases: manual Seismic engineering domain ontology has been created. Then a part of this ontology has been semi automatically generated using the JENA API and Virtuoso for storing Ontology.

This chapter is organized as follows. Section 4.2 provides an overview of DERA methodology providing a set of guidelines for designing ontological conceptual models for standards. Section 4.3 defines the representation of seismic engineering ontology in RDF; Section 4.4 presents the ontology integration approach; Mappings used to connect ontologies to information sources and mappings are the topic of section 4.5; Section 4.6 describe ontology alignment techniques with large lexical database named WordNet; while section 4.7 7 contains ontology evolution approaches with their contribution. Finally, we summarize this chapter.

4.2 Ontology Development Methodology

The DERA methodology defines a systematic approach of SEPREMO ontology development that is scalable and extendable, this approach was used in developing different ontologies such as GeoWordNet (Giunchiglia et al. , 2010). Moreover, SEPREMO represent a set of concepts within a domain and the relationships between those concepts.

DERA methodology, a faceted approach, allows building domain specific ontologies. Domain based ontology is a set of concepts, relations and attributes that specify shared knowledge concerning target domain. For example, earthquake engineering, oceanography, medicine, mathematics and computer science can be considered as separate domains. In DERA, a domain is composed of three- tuple ***D =*** $<$***E, R, A*** $>$.

To do conceptual analysis and knowledge representation; this thesis will be addressed as domain ontology. Among the macro-steps to develop each component of a domain ontology, we used the following ones.

In the first step (identification) towards building an ontology, we identified the atomic concepts of terms collected from research RELUIS database for the earthquake en-

gineering research community, papers, books, existing ontological resources and experts belonging to Earthquake Engineering domain giving emphasis on research projects and experiments aspects. It is an important step to minimize the amount of data and concepts to be analyzed, especially for the magnitude and complexity of the budgetary semantics. In successive iterations for verification process, it will be adjusted if necessary. The collection of candidate terms usually focuses on the identification of noun phrases (NP), through the application of NLP techniques for normalization and linguistic processing such as part-of-speech tagging and tokenization. It retrieves all possible terms in the form of single word or multi-word terms. After collecting terms we examined and disambiguated into atomic concepts. We found terms such as device, shaker, experiment, dynamic test, and identified the atomic concept for each of them. We bootstrapped our Knowledge Base with the concepts and relations of WordNet [1].

Terms with same meaning (synonyms) are grouped together and are given a natural language description that makes explicit the intended meaning. This helps scoping the domain and the class hierarchy. This term is then arranged into facets (Dutta et al., 2011). For instance, the term experiment (defined as the act of conducting a controlled test or investigation) is more appropriate than term test. Here we only consider laboratory experiment that means physical experiment. On the other hand test may be performed at laboratory or in the computer system. In facets hierarchy should be classified properly otherwise we will miss the proper relation between parent and child node (Dutta et al., 2011). For example, to classify specification on the document, we need to classify the document like nominal property, device, structural component, specimen, material, project, and experiment. The classification becomes incomplete if we miss any of these terms.

We also consider the relations between instances that can be mapped by part meronym (part-of) relation and relation between class and instances can be mapped to instance hyponym (instance-of) relation. For example, relationship between element and structural component has substance meronym relation and structural component and specimen has a subsumption relationship. The shape of the facets also considers broader and narrower terms. Another important point is that, we avoid plu-

[1] http://wordnet.princeton.edu/

ral terms because this terms refer group of entities e.g. pile(s). Schema mentioned above provides a vocabulary and set of rules for converting terms to a normalized set of concepts rules that provide groups of terms to build proper facets.

In the second step (analysis) we analyzed the concepts, i.e., we studied their characteristics to understand the similarities and differences between them. The main goal is to identify as many distinguishing properties - called characteristics - as possible from the real world objects represented by the concepts. The term device has 5 different concepts in WordNet. In our case, we selected the one that has the following description: device – (an instrumentality invented for a particular purpose). In this fashion, we have found 193 atomic concepts.

Once the analysis was completed, in the third step (synthesis) we organized them into some facets according to their characteristics. For example, shaker is more specific than device, actuator is more specific than device, motor is a part of electric actuator and we assigned the following relationships between them: shaker IS_A device, actuator IS_A device, motor PART_OF electric actuator. This is how we built device facet. In this way, we built 11 facets. A partial list of the facets is as follows: device, experiment, specimen, experimental computation facility, project, project person and organization. Device and experiment facets are shown in Figure 4.1.

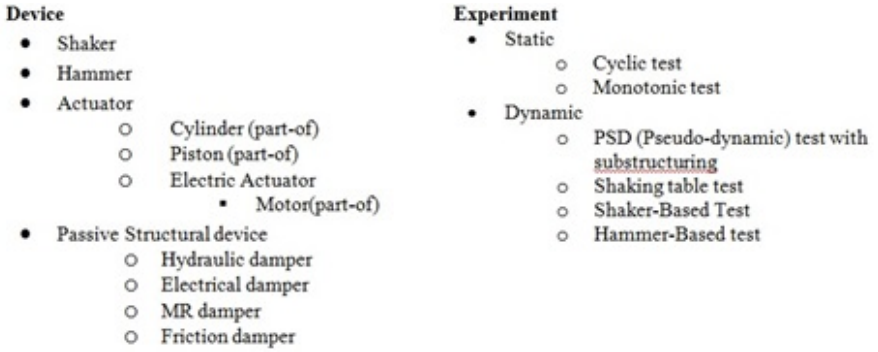


Figure 4.1: The Device and Experiment Facets

Note that in Figure 3.1, concepts which are connected by *PART_OF* relation with the concepts one level above in the hierarchy are explicitly written, for example, motor is *PART_OF* electric actuator. In the other cases, *IS_A* relation holds between them,

for example, electric actuator *IS_A* actuator.

In the fourth step (standardization), we marked concepts with a preferred name in cases of availability of synonymous terms. This approach minimizes the ambiguity through identifying the term which is most commonly used in the domain. WordNet also follow this approach where terms are ranked within synset and the first one is preferred. For example, while experiment and test are used to refer to the same concept, we assigned the former term as the preferred one. This is contrasting from the faceted approach that consider only one term is conserved in the classification while the others are discarded. After that, the ontology was validated by domain experts. Finally we order them according to the importance.

4.3 Ontology Representation

This section describes how the methodologies outlined above have been incorporated in the final design of the RDF language. These statements take the form of subject, predicate, object triples $<$s, p, o $>$a syntactic variant of traditional binary predicates, e.g. p(s, o). The assertion of such a triple is defined to mean that predicate p is a relation between s and o. Each part of the triple, i.e. each RDF name, denotes a resource.

A name is treated depending on its syntactic form on its syntactic form: URI references are treated as logical constants, but plain literals of the form ”literal value” denote themselves and have a fixed meaning. A literal that is typed by an XML Schema datatype; a resource that has a name which is a URI reference, denotes the entity that can be identified by means of the URI. It does not denote the URI itself; nor does it necessarily denote the entity found at the location when the URI is dereferenced as if it were a URL. In other words, a RDF resource can be anything, and does not have to exist on the web. Furthermore, a URI cannot be used to identify multiple entities.

For example, SEPREMO RDF graph is serialized in RDF/OWL language as follows (see Figure 4.2).

```
<?xml version="1.0" encoding="windows-1252"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:ontology="http://earthquake.linkeddata.it/ontology/"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/PassiveDevice">
   <rdfs:subClassOf rdf:resource="http://earthquake.linkeddata.it/resource/Device"/>
   <ontology:desciption rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A device that does not require a
source of energy for its operation.</ontology:desciption>
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
 </rdf:Description>
</rdf>
```

Figure 4.2: SEPREMO RDF graph

RDF/XML uses the rdf:about property to state that some rdf:Description concerns the resource indicated by the URI reference. The rdf:resource property connects the predicate of a relation to its object, e.g. the object of the rdfs:subClassOf relation in the statement Passive Device is a Device. Provided that the type of some resource is known, as is the case with the ontology:description, we can directly state the definition of that resource under an element of its type.

In addition we also represent the information of experimental data in RDF/OWL. We will discuss detail about SEPREMO RDF graph in next section.

4.4 Ontology Integration

To have a Semantic Web system which allows computers to combine and infer implicit knowledge from different ontologies in a particular domain of interest, these ontologies should be linked and related to each other. The primary goal of ontologies is knowledge sharing, so ontologies are often reused and distributed in a large scale. By merging and reusing the ontologies, the system would be more effective for information retrieval, query answering and problem solving.

SEPREMO is an integrated ontology which is using and re-using different accessible domain specific ontologies. By reusing concepts from other generic ontologies, a well-defined concept will be obtained which is easier to share. The reuse of existing ontologies and adapting them for a particular purpose is often not possible without

considerable effort (Uschold et al., 2011).

Developed facets include concepts that were selected from NEES thesaurus to be incorporated into our ontology. In fact this integration was accomplished when we built the facets. The Network for Earthquake Engineering Simulation (NEES) is one of the leading organizations for Earthquake Engineering in USA. They developed the earthquake engineering thesaurus; it is based on Narrower and Broader terms. It contains around 300 concepts and in our ontology we have integrated 75 concepts from NEES. Figure 4.3 depicts a small portion of NEES thesaurus.

NEES Earthquake Engineering Ontology by Broader Term		
Broader Term	Term	Narrower Term
AASHTO_2001	AASHTO_LRFD_Bridge_Design_Specifications	
Acceleration	Peak_Base_Acceleration	
Actuator	Dynamic_Actuator	
Actuator	Static_Actuator	
Axial_Load	Cyclic_Axial_Load	
Bearings	Preformed_Fabric_Pads	Cotton_Duck_Bearing_Pads

Figure 4.3: NEES Thesaurus

In this Section, we describe how we integrated our developed ontology with Wordnet. Basically, we applied the semi-automatic ontology integration algorithm proposed in (Farazi et al., 2011). In particular, we implemented the following macro steps:

a. Concept Integration

1. ***Facet concept identification***: For each facet, the concept of its root node is manually mapped to WordNet, in case of availability.

2. ***Concept Identification***: For each atomic concept C of the faceted ontology, it checks if the concept label is available in WordNet. In case of availability, it retrieves all the concepts connected to it and maps with the one residing in the sub-tree rooted at the concept that corresponds to the facet root concept. We restrict to noun senses only.

3. ***Parent Identification***: In case of unavailability of a concept it tries to identify parent. For each multiword concept label it checks the presence of the header, and if it is found within the given facet, it identifies it as a parent. For instance, in WordNet

it does not find hydraulic damper for which damper is the header and that is available there in the hierarchy of device facet. Therefore, it recognizes the damper with the description damper, muffler – (a device that decreases the amplitude of electronic, mechanical, acoustical, or aerodynamic oscillations) as the parent of the hydraulic damper.

b. Instance Integration

WordNet, the specific instance hypernym relation is used to link a synset denoting an entity to the synset denoting the corresponding class (or classes). To count this point, we introduced a new object in the entity part of our knowledge base that distinguish between concepts and instances. We also created part meronym relations between such entities, according to the information provided in SEPREMO.

Moreover, we use inference algorithms extract implicit knowledge from a given knowledge base. Standard reasoning tasks include instance integration, consistency checks and subsumption.

c. Metadata Importing

Experiment in SEPREMO contains some metadata including organization name that performed the experiment, experiment name, computation type, repetition, loading name, loading coefficient, peak excitation. For instance, Nominal loading is (e.g., 100%) and Peak Excitation is the effective magnitude of the loading (for instance 0.01m or 0.20 g) depending on the type of experiment. In this case we said this experiment may be identified as static or pseudo-dynamic. We attached all information to the corresponding object created for the earthquake engineering project entity in the entity part of knowledge base.

4.5 Ontology Mapping

Ontology mapping is an important step to achieve knowledge sharing and semantic interoperable in an environment in which knowledge and information have been represented with different ontologies. The process of ontology mapping species the semantic overlap between two ontologies. Furthermore, one closely related research

topic with ontology mapping is schema matching, which has been one major area of database research (Doan et al., 2003).

Mapping two ontologies, O1 onto O2, means that each entity in ontology O1 is trying to find a matching entity which has the same intended meaning in ontology O2 (Giunchiglia et al., 2005). This algorithm is based on a combination of methods which uses the definition of the concept and its structure. The definition of the concept is the main consideration when mapping the concept of an ontology based on names, descriptions and relations; the conceptual structure method considers the concept of hierarchy among areas such as the relationship between nodes (parent node, sub-node) and semantic relations between neighbors.

The first issue, we mapped SEPREMO to WordNet and DBpedia. Note that the official number of entities in WordNet is 7671 (Miller and Hristea, 2006), while we found out that 683 of them are common nouns. We only consider synset classes, the attributes and the nearby relation. We address the meaning of similarity between two concepts. Clearly, many different definitions of similarity are possible, each being appropriate for certain situation. In this case, ontology mapping is used to map a concept found in SEPREMO, or a query over WordNet if they denote the same meaning. We also consider partial match if there is a corresponding synset in WordNet but the word in the SEPREMO synset is not present in the WordNet synset. This would be mean that, for example, test, experiment variants belong to the same synset.

The second challenge is then to find a more general synset according to the IS_A (hypernym)relation considering match case in our ontology. This challenge can form the backbone of a knowledge base or lexicon, via which rich semantic specifications can be inherited in a consistent way to thousand so more specific concepts. In SEPREMO, we have tried to encode multiple hypernym relations more comprehensively. However, hierarchical structures quickly become very complex once this is allowed and consistency should be checked by actually implementing and applying inheritance. Consider for instance the class Pseudo-dynamic test, defined in SEPREMO as An experiment which is a simultaneous simulation and control process in which inertia and damping properties are simulated and stiffness properties are acquired from the structure. We found that there is no equivalent synset for it in

WordNet, but the more general synset for experiment, defined as An empirical method that arbitrates between competing models or hypotheses is available in WordNet. So, Pseudo-dynamic test in SEPREMO is marked as more specific than experiment in WordNet.

In SEPREMO, the complex relations are needed to help the relation assignment during the development process when there is a lexical gap in one language or when meanings do not exactly fit . To consider this point, finally we consider the part-of (part meronym) relation instead of the is-a relation. The meronymic relations transitive (with qualifications) and asymmetrical (Cruse, 1986), and can be used to construct a part hierarchy (with some reservations, since a meronym can have many holonyms). For example, in our experiments, meronym candidates are (cylinder and piston) pairs for the actuator class of SEPREMO.

4.6 Ontology Alignment

Ontologies must be available for sharing or reusing; therefore, semantic heterogeneity and structural differences need to be resolved among ontologies. This can be done, by aligning heterogeneous ontologies. Thus, establishing the relationships between terms in the different ontologies is needed throughout ontology alignment. Ontology alignment is the process where for each entity in one ontology we try to find a corresponding entity in the second ontology with the same or the closest meaning. The main goal of the work is to introduce a method for finding semantic correspondences among heterogeneous ontologies, with the intention of supporting interoperability over given domains (SEPREMO).

Alignment systems may also be different in use of external resources in their matching processes such as web resources, external ontologies, dictionaries or semantic resources like WordNet and more on. This section discusses various alignment techniques and specifically those which are used in the SEPREMO to map two entities from different ontologies. Moreover, we pointed out that any ontology alignment technique is not adequate enough to give an accurate match between two entities and hence they are used as a combination of two or more, depending on the algorithm

used in alignment system. The lexical similarity techniques may consider the entity name or label as sequence of characters, string or word as a whole. The combination of structural and lexical matching techniques gives much better idea about the overall similarity of a concept defined in ontology. SEPREMO utilized the results of various alignment techniques which include string-based, linguistic-based and structure-based similarities. Following discussed above mentioned strategies:

1. String Based Strategies

In string-based similarity calculation the entities are considered as strings, regardless of their structures or other associated properties defined in ontology. The string normalization process is made after the basic comparison of entity names. Both entity strings are converted to lower-case and punctuations, dashes and blank character are eliminated. The normalization process play important role in string comparison techniques. For example, Cyclic Test, Cyclic-Test and Cyclic test are normalized to Cyclictest. There is a variety of techniques proposed to calculate the string similarities depending on characteristics of measurements. These techniques include sub-string distance (Euzenat and Shvaiko , 2007).

To do so, various entity categories are taken (classes, properties and instances) of each ontology and divided into separate lists; then the classes from the first ontology are compared with classes from the second: properties vs. properties and instances vs. instances. If the similarity values of the comparison are greater than a predefined threshold, then inserting an element in the matrix with their degree of similarity is essential.

2. Linguistic based Strategies

Linguistic similarities are computed using external resources like language dictionaries for example WordNet, thesauri or specific databases for example RELUIS database. Such similarities are very useful when string-based similarities are not easy to find between entities and it happens when synonyms are used for the same concept in ontologies. For example, the names experiment and test refer to the same concept but the string-based alignment between them is low enough to be ruled out for selection as an alignment candidate. The WordNet is a similar kind of lexical database which provides a repository of lexical items defined as set of semantic vo-

cabulary. In WordNet, different meanings of the same concept are grouped together as sets of synonyms (synsets) in terms of nouns verbs, adjectives and adverbs. In hierarchical manner, synsets are interlinked by means of various conceptual semantic and lexical relations. For example, nouns have relationships of hypernym, hyponym, holonym, meronym and coordinate term. In an SEPREMO, property such owl:equivalentClass could be used to show that entities are same.

The structural similarity information plays vital role in situation where the linguistic or string based similarity between two entities proved to be insufficient or incomplete. This information between two entities comes from their structural features like, their relation with other entities and their direct properties. The main intuitions are given below:

- If two classes from different ontology have similar upper-classes in hierarchy, it is likely that they define the same concept.
- If two classes from different ontology have similar sub-classes in hierarchy, it is likely that they define the same concept.
- If two classes from different ontology have similar properties, it is likely that they define the same concept.
- Two entities having any combination of two or all the three above mentioned similarities suggest more likelihood to be the similar concept.

3. Heuristic based Strategies

Heuristic-based Strategies combine several features of the string matcher with those of iterations, computing the similarities in order to achieve high-quality results. This technique begins by comparing class names, property names and instance by using an editing distance and substring distance between the entity names. In fact, this matcher can work alone and provide a very good result, because it contains all the components of the system.

In this thesis we concentrate only on the first two strategies of the ontology integration phases, leaving the third and fourth phases for future work.

4.7 Ontology Evolution

Ontology evolution can be defined as the process of modifying an ontology in response to a certain change in the domain or its conceptualization (Flouris et al., 2008):

- changes in the domain, when new concepts belonging to the domain are added to reflect new knowledge or a re-purposing of the ontology.
- changes in conceptualization, which can result from a changing view of the domain and from a change in usage perspective

Ontology evolution could be considered as the purest type of ontology change, in the sense that it deals with the changes themselves. Ontology evolution is a very important problem, as the effectiveness of an ontology based application heavily depends on the quality of the conceptualization of the domain by the underlying ontology.

As already stated, an ontology is, according to (Gruber , 1993),), a specification of a shared conceptualization of a domain. Thus, a change may be caused by either a change in the domain, a change in the conceptualization or a change in the specification (Klein and Fensel, 2003). The third type of change (change in the specification) refers to a change in the way the conceptualization is formally recorded, i.e., a change in the representation language. This type of change is dealt with in the field of ontology translation. Thus, our evolution approach covers only the challenges occurred from the changes in conceptualization. The conceptualization of the domain may change for several reasons, including a new observation or measurement, a change in the viewpoint or usage of the ontology, newly-gained access to information that was previously unknown, classified or otherwise unavailable and so on.

In order to manage the complexity of the problem, six phases of ontology evolution have been identified, occurring in a cyclic loop (Stojanovic et al., 2003). Initially, we have the change capturing phase, where the changes to be performed are identified. Three types of change capturing have been identified: structure-driven, usage-driven and data-driven (Haase and Sure, 2004).Once the changes have been determined,

they have to be properly represented in a suitable format during the change representation phase. The third phase is the semantics of change phase, in which the effects of the change to the ontology itself are identified; during this phase, possible problems that might be caused in the ontology by these changes are also determined and resolved. If this were left to an ontology engineer, the evolution process would be too error-prone and time consuming it is unrealistic to expect that humans will be able to comprehend entire ontology and interdependencies in it. This requirement is especially hard to fulfil if the rationale behind domain conceptualization is ambiguous or if the domain experts does not have the experience. For example, when a concept from the middle of the hierarchy is being deleted, all sub concepts may either be deleted or reconnected to other concepts (Breche and Woerner, 1995). If sub concepts are preserved, then properties of the deleted concept may be propagated, its instances distributed, etc. Different ways for resolving the request for the removal of the concept vibrate by considering only the concept hierarchy as shown in the Figure 4.4.

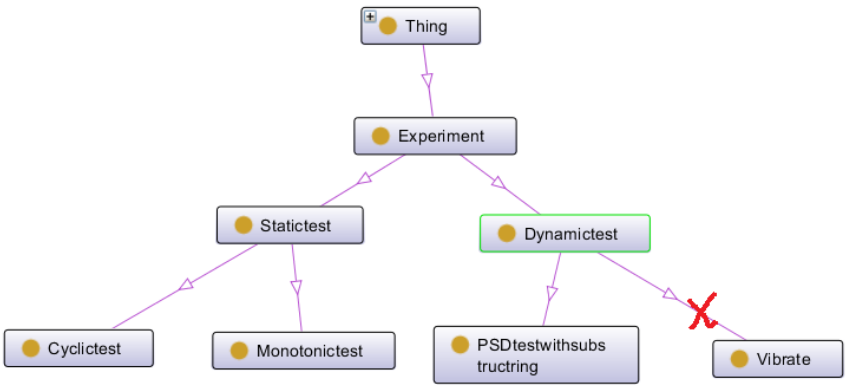


Figure 4.4: After applying the removal of the concept Vibrate.

Moreover, Domain experts suggested a number of changes, e.g., the inclusion of the concepts shaker-based test and hammer-based test in the experiment facet as given below (Figure 4.5):

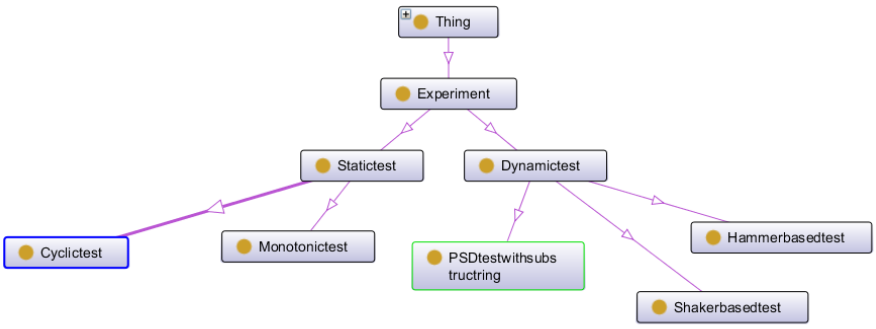


Figure 4.5: Inclusion of the shaker-based test and hammer-based test in the experiment facet

The change implementation phase follows, where the changes are physically applied to the ontology, the ontology engineer is informed on the changes and the performed changes are logged and six log files in our developed ontology. These changes need to be propagated to dependent elements; this is the role of the change propagation phase. Under that viewpoint, ontology evolution is concerned with the ability to change the ontology without losing data or negating the validity of the ontology, while ontology versioning should additionally allow access to different variants of the ontology. Ontology evolution is concerned with the validity of the newest version; ontology versioning additionally deals with the validity, interoperability and management of all previous versions, including the current one (Stojanovic et al., 2003), which is directly affected by the ability of an evolution algorithm to properly adapt the ontology to changes in the domain and to new needs in the conceptualization.

Finally, the change validation phase allows the domain experts to review the changes and possibly suggest variations, if desired. This phase may uncover further problems with the ontology, thus initiating new changes that need to be performed to improve the conceptualization; in this case, we need to start over by applying the change capturing phase of a new evolution process, closing the cyclic loop. An alternative, but similar, approach which identifies five phases, can be found in (Plessers and Troyer, 2005).

4.8 Conclusion

In this chapter we have introduced and followed the large-scale ontology design and development case study in the domain of Earthquake engineering. We followed DERA methodology for building this domain specific ontology. We studied which studied resources to find out which one to use in our tasks and how to use each of them, how to discover instances from those resources how to link their content and understand the content of the resources and interpret the results. After that, ontology representation language was presented. There are many ontology languages; we chose the RDF/OWL language for our approach since it overcomes the defects appeared in other ontology languages. We exploited an ontology integration algorithm that was employed to incorporate our ontology into WordNet. It helped to increase the coverage of the Knowledge Base. We have also presented a technique for ontology mapping. We formalized the notion of ontology, ontology morphism and ontology mapping and linked them to the WordNet and DBpedia. Moreover, we showed that ontology alignment technique that is essential to the development of applications that leverage the potential of the Semantic Web. Most current state of the art alignment systems are capable of identifying only the simplest of relationships between ontologies: 1-to-1 equivalence. Ontology alignment section also showed that the string preprocessing strategies, such as stop word removal; Linguistic based strategies for example consideration of synonyms; structure similarities and Heuristic based Strategies. Finally, we represent a novel approach for dealing with ontology evolution. The approach is based on a six-phase evolution process, which systematically analyses the causes and the consequences of the changes and ensures the consistency of the ontology and depending artefacts after resolving these changes.

CHAPTER 5

ONTOLOGY PUBLISHING

5.1 Introduction

The vision of the ontology development is that of a world-wide distributed architecture where data and services easily interoperate. This vision is not yet a reality in the Web of today, in which given a particular need, it is difficult to find an earthquake engineering resource that is fit for the user queries. Also, given a relevant resource, it is not easy to understand what it provides and how to use it. To solve such limitations, facilitate interoperability, and thereby enable the ontology vision, the key idea is to publish semantics descriptions of Web resources for example RDF. These descriptions rely on semantic annotations, typically on logical assertions that relate resources to some terms in predefined ontologies. Moreover, this chapter also shows the procedure for publishing earthquake engineering vocabularies.

5.2 Vocabularies

The automatic integration of information resources in the earthquake engineering is one of the most challenging goals for earthquake engineering research projects and experiments today. Controlled vocabularies have played an important role in realizing this goal, by making it possible to draw together information from heterogeneous sources secure in the knowledge that the same terms will also represent the same entities on all occasions of use. We use knowledge acquisition techniques with manual terminology extraction and a final review is provided by domain experts to validate

acquired knowledge.

A vocabulary contains the fundamental building blocks used to lead complex thoughts, including physical objects, abstract ideas, their properties, and their relationships. A basic unit of a vocabulary is the term, defined as a lexeme used in a particular domain, that is, the basic linguistic units, composed of form and meaning (or concepts). The word term can have three common senses(Crystal , 1980):

- Word-Form: An entity or physical object found in written and spoken text
- Lexeme: An abstraction that expresses a set of grammatical variants (e.g., think, thinks, thinking, and thought)
- Word: An abstraction that functions as a fundamental building block of grammar.

Developing Vocabulary typically refers to the process of creating a controlled vocabulary, defined as a way to represent thesaurus of canonical terms for describing every concept in a domain. In this thesis, generating a vocabulary refers to collecting and organizing a set of terms representative of a vocabulary assumed to exist. For example, we consider SEPREMO to include all terms that earthquake engineering research community use to discuss earthquake engineering related projects and experiments topics. Thus, generating the SEPREMO is shorthand for creating a representative set of words based on specified criteria. Creating a vocabulary involves extracting terms that describe domain-specific concepts or entities from relevant sources of discourse, such as collections of documents, interviews of domain experts, and RELUIS database. The level of specificity depends on the type of vocabulary and its purpose. SEPREMO uses faceted based approach to specify concepts, relations and specify attributes.

Every concept must have an unique identifier that remains unique and constant in meaning. In SEPREMO, the unique identifier is a meaningful word that people use to denote a concept may change over time, but the concept itself does not change. A concept name should be as explicit and as unambiguous as possible. To acquire earthquake engineering research projects and experiments knowledge used by associate researcher we applied a procedure which is divided into two main steps: the first aims at the identification of earthquake engineering terms related to experiment and

project; the second step consists of the acquisition not only of experiment and project, but also of terms related to experiment procedure, people involved in the project, institutes involved in the project and devices and specimen use for experiment.

Synonyms are also considered while developing CV and sometimes assigned to concepts assigned to concepts. A synonym is an alternate name for the concept. Synonyms help users to search for concepts; therefore, near-synonyms are permitted. Abbreviations are similar to synonyms in that, which are used to facilitate search. However, they are distinct from synonyms and maintained in a separate data structure. A synonym or abbreviation may be used for two different concepts. For example, experiment is a synonym for test. Moreover, a concept definition is optional but desirable. A text definition is presented in structured natural language, like dictionary definition. Following figure 5.1 represents subset of entity classes of the SEPREMO ontology:

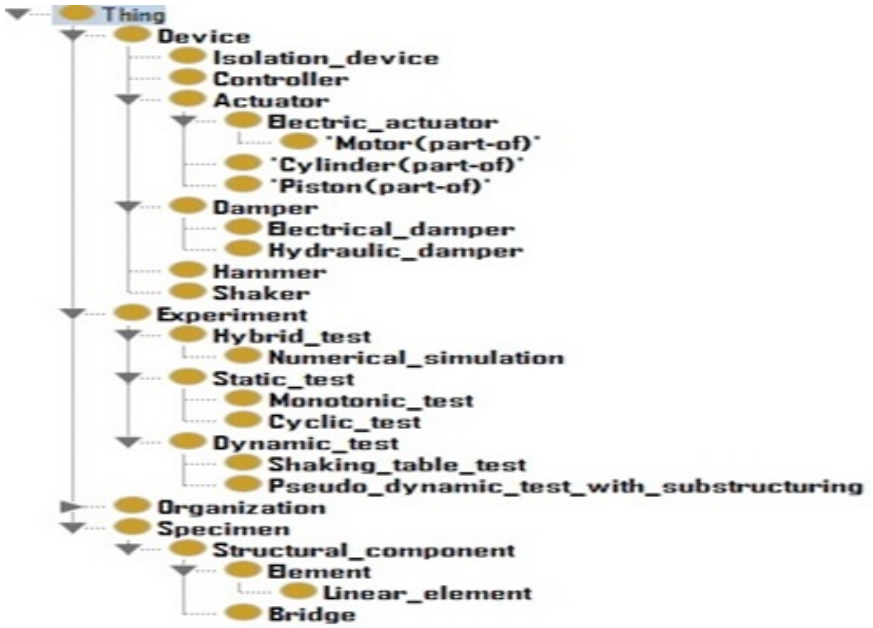


Figure 5.1: A subset of the entity class concepts of the SEPREMO

In Figure 5.1,a subsumption hierarchy is defined in SEPREMO by the specification of parents and children. The relationship between parents and children is always is-a.

For example, device is a parent of actuator, and electric actuator is a child of actuator. Although children could be inferred from parents, SEPREMO includes children in the concept model to facilitate more efficient retrieval of hierarchical information by an implemented system. Subsumption relationships are inherited down the hierarchy (i.e., subsumption is transitive). Ancestors and descendants are not explicitly included in the concept model, because, when necessary, ancestors and descendants can be computed recursively from parents and children by the implemented system.

Attribute facet is a set of attribute value pairs that define the concept. Attribute value pairs are inherited down the hierarchy; the values can be restricted further at lower levels of the hierarchy. The attribute value pairs should contain information about the concept that is established. SEPREMO attribute has a code that serves as an attribute unique identifier, and that remains constant in meaning over time. It also has an attribute name that is unique at any given moment, but that may change over time. Following presents attribute facet of SEPREMO.

Figure 5.2: A partial list of attribute concepts added to the SEPREMO

Whenever experiments are returned through the SEPREMO ontology, attributes can be set so that the result contains a list of experiment name, computation type, repetition, loading name, loading coefficient, peak excitation, all of which contain information regarding the type of experiment done available per facet.

Relation facet presents intra-facet relationship; because all the terms within a facet come into the same category. If a relationship is mixed in a single vocabulary, the relationship should be flagged for clarity. Relationships between terms from different

facets are in de facto associative relationships. The difference between Relations and Attributes boils down to the nature fillers: Relations have references to concepts in their range slots; Range slots of attributes can contain elements from specific value sets (Sergei and Raskin , 2004).Following table represents relation between facets in the SEPREMO:

Facet	**Relation**	**Facet**
Project Person	Works_in	Project
Project	Use	Experimental Computation Facility
Experiment	Generate	Computer File
Project	Use_resources	Organization
Experiment	Belongs_to	Project
Specimen	Belongs_to	Project
Experiment	Use	Specimen
Experiment	Use	Specimen
Experiment	Generate	Specification
Specification	Follow	Project template

Table 5.1: A partial list of relation concepts added to the SEPREMO

For instance, for a given specimen, there is a unique configuration that can be used for all experiments of all projects using this specimen.

5.3 Web Ontology Languages

In the following subsections, we describe the Knowledge Representation Languages RDF, RDFS and OWL in terms of their capacity in representing ontologies of varied kinds.

5.3.1 Resource Description Language (RDF)

The formalized ontology language provides a possibility for users to describe concepts of domain model explicitly and formally. Therefore, it should meet the follow-

ing requirements: a well-defined syntax, a well-defined semantic, efficient reasoning support, sufficient expressive power, convenience of expression. The domain specific ontology was published into RDF by means of Jena (a Semantic Web tool for publishing and managing ontologies) and integrated with WordNet RDF using the approach described in chapter 3. In the next section author will discuss how to publish ontology using JENA API.

To generate the RDF model of our SEPREMO we created a JENA API taking the plain text file created in excel as input which is result of the term extraction process and which resulted from the term extraction process and also from manual review by domain experts. So, RDF graph is stored in Jena as a model, and a Jena model is created by a factory, as in:

Model m=ModelFactory.createDefaultModel();

Once a model has been defined, Jena can populate it by reading data from files for example Excel, backend databases. in various formats and once it has been populated, Jena can perform set operations on pairs of populated models and /or search models for specific values or combinations (patterns) of values. Figure 4.1 reports a small portion of the input file.

A	B	C	D	E
Parent	**Parent_Description**	**Child**	**Child_Description**	**Relationship**
Device	An instrument used for a particular purpose.	Hammer	A hand tool with a heavy rigid head and a handle; used to deliver an impulsive force by striking	Is_A
Device	An instrument used for a particular purpose.	Controller	A mechanism that controls the operation of a machine	Is_A
Device	An instrument used for a particular purpose.	Potentiometer	A device for measuring direct current electromotive forces.	Is_A
Device	An instrument used for a particular purpose.	Actuator	A mechanical device for moving or controlling something.	Is_A
Actuator	A mechanical device for moving or controlling something.	Dynamic Actuator	An actuator based on dynamic applications and the maximum operating conditions such as operating frequency and signal amplitude	Is_A
Actuator	An instrument used for a particular purpose.	Static Actuator	A actuator based on the available stroke(strain), hysteresis, stiffness and load capability	Is_A
Device	An instrument used for a particular purpose.	Active Structural Device	A vibration control device that incorporates real time recording instrumentation on the ground.	Is_A
Device	An instrument used for a particular purpose.	Passive structural Device	A vibration control device that does not incorporate feedback receiving capability from the devices of this kind, structural elements a	Is_A
Device	An instrument used for a particular purpose.	Damper	A device that decreases the amplitude of electronic, mechanical, acoustical or aerodynamic oscillations	Is_A
Damper	A device that decreases the amplitude of electronic, mecha	Magneto rheological damper	A damper filled with magnetorheological fluid, which is controlled by a magnetic field, usually using an electromagnet	Is_A
Damper	A device that decreases the amplitude of electronic, mecha	Hydraulic Damper	A damper that converts the kinetic energy of moving components into thermal energy	Is_A
Damper	A device that decreases the amplitude of electronic, mecha	Electrical damper	A damper that is commonly used as a ventilation	Is_A
Damper	A device that decreases the amplitude of electronic, mecha	Friction damper	A damper that is used to reduce squeal noise.	Is_A
Damper	A device that decreases the amplitude of electronic, mecha	Tuned mass damper	A damper mounted in structures to reduce the amplitude of mechanical vibrations	Is_A
Damper	A device that decreases the amplitude of electronic, mecha	Elastomeric Damper	A damper that works as an energy dissipating components.	Is_A
Damper	A device that decreases the amplitude of electronic, mecha	Isolator	A device which is a knife switch, designed to open the circuit under no load condition and can be used as a Seismic Isolator.	Is_A
Isolator	A device which is a knife switch, designed to open the circu	High damping Isolator	An isolator with good load support ability, restoring force and damping force.	Is_A
Isolator	A device which is a knife switch, designed to open the circu	Seismic Base Isolator	An isolator that used a technique developed to prevent or minimize damage to buildings during an earthquake.	Is_A
Damper	A device that decreases the amplitude of electronic, mecha	Viscoelastic damper	A damper consisting of viscoelastic material bonded to steel plates to reduce seismic and wind responses.	Is_A
Damper	A device that decreases the amplitude of electronic, mecha	Metallic damper	A damper that mainly consists of low carbon steel rods; that can be installed between the superstructure and the substructure of a st	Is_A

Figure 5.3: Excel view of SEPREMO

As shown in the Figure 5.3, SEPREMO is composed of several columns that on one hand represents the earthquake engineering category that each term belongs to (e.g. device, project person, damper, etc.), and on the other hand it represents various attributes associated with each term (e.g. Parent_Description,Child_Description, Relationship). In the process of converting to RDF each term was translated into

a class of the RDF model and each category into a superclass, while attributes associated with the terms became properties. In particular, column A (Parent) and C (Child) represents the main classes of the RDF model. Moreover, each class from column A is a subclass of the corresponding class in the column C. For example, the term Hammer in RDF is a subclass of the category Device. In the resulting triples, the subject is most often the SEPREMO concept. The predicates correspond to concept properties, which include type (concept or relationship), preferred name (label), and relations to other concepts (e.g., subClassOf). The following RDF statements describe the resources Specification, DisplacmentSensor, Frictionpendulumbearing, ConductivitySensor, Isolator and ExperimentalComputationalFaclity.

```
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/DisplacementSensor">
  <ontology:desciption rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A sensor capable of high-resolution measurement of the position
  and/or change of position of any conductive target</ontology:desciption>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/Frictionpendulumbearing">
  <ontology:desciption rdf:datatype="http://www.w3.org/2001/XMLSchema#string">An isolation device that  has the characteristic of a pendulum to
  lengthen the natural period of the isolated structure to avoid the strongest earthquake force.</ontology:desciption>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>

<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/ConductivitySensor">
  <ontology:desciption rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A sensor which measures thermal conductivity of air due to humidity.</ontology:desciption>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/Document">
  <rdfs:subClassOf rdf:resource="http://earthquake.linkeddata.it/resource/Specification"/>
  <ontology:desciption rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A computer file that contains text and possibly formatting instructions.</ontology:desciption>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/Isolator">
  <ontology:desciption rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A device which is a knife switch, designed to open the circuit under no load condition
  and can be used as a Seismic Isolator.</ontology:desciption>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/ExperimentalComputationFacility">
  <ontology:desciption rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A computational facility for the experiments.</ontology:desciption>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
```

Figure 5.4: RDF describes the Resources

The key RDF package for the application developer is com.hp.hpl.jena.rdf.model. This API has been defined in terms of interfaces so that application code can work with different implementations without requiring any change. This package contains interfaces for representing models, resources, properties, literals, statements and all the other key concepts of RDF, and a ModelFactory for creating models. So the application code remains independent of the implementation, it is best if it uses interfaces wherever possible, not specific class implementations.

As we mentioned in vocabulary construction SEPREMO has an attribute facet. Figure 5.5 depicts how experimental data from experimental result files was published in RDF.

```
<rdf:Description rdf:about="http://earthquake.engineering.unitn.it/resource/Experiment1">
    <rdf:type rdf:resource="http://earthquake.engineering.unitn.it/resource/Experiment"/>
    <earthquakeontology:load_displacement>(0.62 0.52, 338.81 61.28, 375.41 122.02, 411.45 182.77)
    </earthquakeontology:load_displacement>
</rdf:Description>
```

Figure 5.5: A snippet of the experimental data represented in RDF

New terms were created only in case suitable candidates were not available in the standard vocabularies. Notably we have created load_displacement terms based on load and displacement of the experiment. Despite the fact that the earthquake engineering community is nontrivially contributing to the Cloud, finding datasets for experiments such as dynamic tests, pseudo-dynamic tests and cyclic tests is a far cry from what it has been expected. In the opinion of the author that publishing such experimental data has largely been overlooked and, as such, to the best of our knowledge no vocabulary is yet developed in this field, to model data in RDF (Hasan et al. , 2004).

5.3.2 RDF Schema

As described in the RDF section, RDF builds upon the notion of resources, information units that can have certain properties with corresponding values. In turn, modeling elements of the RDF language are also resources. RDF schema refines the notion of modeling resources. RDF schema defines standard properties, constraint properties and classes. Figure 5.6 gives an overview of the introduced resources that will be discussed in the following section.

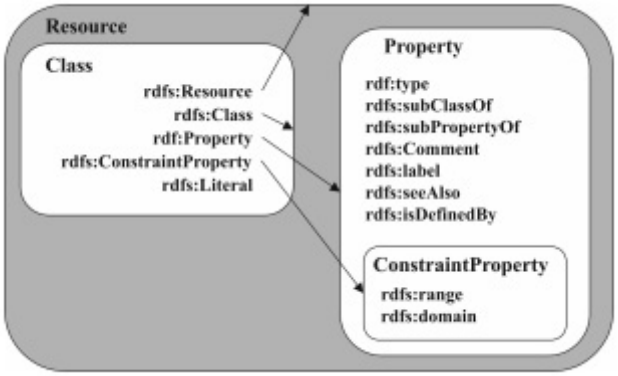


Figure 5.6: Modeling Components of RDF schema from (Brickley et al. , 2004)

While the RDF language contains the rdf:type operator, there is no explicit notion of classes. RDF schema fills this gap by introducing classes as special kinds of resources. They are identified by the resource rdfs:Class. A general resource can be identified as a class using the rdf:type property. RDF schema also defines the rdfs:subClassOf property for specifying hierarchies. In our example, we could define represented relationship between Hammer and Device concept; and the rdfs:subClassOf property is used to relate the former class to its more generic class generated later . RDF schema allows multiple inheritances. Thus we can define a mother to be a subclass of parent as well as female person(Figure 5.7):

```
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/Hammer">
        <rdfs:subClassOf rdf:resource="http://earthquake.linkeddata.it/resource/Device"/>
        <ontology:desciption rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A hand tool with a heavy rigid head and a
        handle; used to deliver an impulsive force by striking</ontology:desciption>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>

    <rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/Damper">
        <rdfs:subClassOf rdf:resource="http://earthquake.linkeddata.it/resource/Device"/>
            <ontology:desciption rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A device that decreases the amplitude of
        electronic, mechanical, acoustical or aerodynamic oscillations</ontology:desciption>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    </rdf:Description>
```

Figure 5.7: RDFS describes the Resources ”Hammer” and ”Damper”

The main descriptive element of RDF are properties of resources specified by the RDF resource rdf:label specified in the RDF name space. rdfs:label is an instance of rdf:Property that may be used to provide a human-readable version of a resource’s

name. Properties are used to describe arbitrary binary relations between resources. For example we can describe the following relations(Figure 5.8) using RDF properties,which is in our domain:

```
<rdf:Description rdf:about="http://earthquake.engineering.unitn.it/resource/Specimen">
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Measuring hardness</ontology:description>
  <rdfs:label>mean_property</rdfs:label>
  <rdfs:label>scaled_property_name</rdfs:label>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Average strength of the specimen</ontology:description>
  <rdfs:label>structural_component</rdfs:label>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Length of the Specimen</ontology:description>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Structrual Compoent in specimen</ontology:description>
  <rdfs:label>max_width</rdfs:label>
  <rdfs:label>max_height</rdfs:label>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Depth of the Specimen</ontology:description>
  <rdfs:label>mass</rdfs:label>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Mass of the Specimen</ontology:description>
  <rdfs:label>max_depth</rdfs:label>
  <rdfs:label>max_length</rdfs:label>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Width of the Specimen</ontology:description>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Height of the Specimen</ontology:description>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.engineering.unitn.it/resource/Sensor">
  <rdfs:label>Location</rdfs:label>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Where the sensor located in the Specimen</ontology:description>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.engineering.unitn.it/resource/Device">
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">giving quantitative measurments</ontology:description>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Detail about configuration of device</ontology:description>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string"></ontology:description>
  <rdfs:label>calibration</rdfs:label>
  <rdfs:label>stroke_capacity_value</rdfs:label>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Name of the device</ontology:description>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Includes all the activities and techniques of maintaining the stocks of items</ontology:description>
  <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Name of the company who made device</ontology:description>
  <rdfs:label>inventory_status</rdfs:label>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  <rdfs:label>device_name</rdfs:label>
  <rdfs:label>Device_producer_name</rdfs:label>
  <rdfs:label>device_configuration</rdfs:label>
  <rdfs:label>force_capacity_value</rdfs:label>
</rdf:Description>
```

Figure 5.8: Attribute presents in RDF

Using these properties, we can describe members of the SEPREMO as resources and relate them by the usual relation like rdfs:label and ontology:description is the Literal for the specific resources. RDF schema now defines the special property rdfs:label that can be used to define a specialization of an existing property.

Finally, RDF schema provide a controlled vocabulary for specifying the terminological structure of a domain with a semantics that can be implemented in a formal logic in order to provide simple inference services like type checking or reasoning. However, it has limitations; for example, it cannot be used to define whether a property is symmetric or transitive. To model such axioms, W3C introduce ontologies Web ontology language (OWL).

5.3.3 Web Ontology Language (OWL)

OWL can facilitate more precise ontology description than RDFS. The OWL specifications contain many features and capabilities that are useful to describe Web ontologies. For example, while using OWL, ontology can explicitly describe more precisely.

Overall, OWL was invented to utilize XML syntax and to adopt RDF and RDFS primitives; for example, it uses RDF terms and meaning in defining classes and properties. Moreover, OWL is based on DL that formally describes the meanings of terminologies used in web documents. It was also designed to overcome RDF weaknesses.

OWL is an emerging language to represent ontologies in semantic web and recommended by World Wide Web (WWW). As its vocabulary is used to describe the semantics of ontology, it can also be used to find some indications for matching entities during the ontology alignment process. In Figure 5.9, we present a part of the OWL syntax, for example, owl:Class rdf:about="Experiment" is used to define a class and its name is Experiment. Similarly, the syntax rdfs:subClassOf defines a class which is a sub-class of another defined class in ontology. The rdfs:subClassOf construct is defined as part of RDF Schema. This property is transitive for example if query asks for all Device, also resources that are classified as Hammer, Damper should be returned. Therefore properties associated with a superclass also apply to subclass. The rdfs:subClassOf property can be used with a class and its value must be a class or property restriction.

The owl:equivalentClass is used to define two class descriptions involved who have same class extension i.e., both class extensions contain exactly the same set of individuals for example Experiment and Test are equivalent. The owl:equivalentClass property is used to identify a synonymous class. Two classes are equivalent if and only if they are subclasses of each other. The simplest form of specifying the equivalence of classes is to use their names.

```
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/PlaneElement">
    <owl:equivalentClass rdf:resource="http://earthquake.linkeddata.it/resource/2DElement"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/Experiment">
    <owl:equivalentClass rdf:resource="http://earthquake.linkeddata.it/resource/Test"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/ExperimentalComputationalSystem">
    <owl:equivalentClass rdf:resource="http://earthquake.linkeddata.it/resource/DataAcquisitionSystem"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/StrainGauge">
    <rdfs:subClassOf rdf:resource="http://earthquake.linkeddata.it/resource/Sensor"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A sensor for measuring strain in a surface.</ontology:description>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/SonicProfileTransducer">
    <rdfs:subClassOf rdf:resource="http://earthquake.linkeddata.it/resource/Sensor"/>
    <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A sensor used for contactless fill level measurement, flow rate measurement,
    distance measurement, object detection and automation.</ontology:description>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/Controller">
    <rdfs:subClassOf rdf:resource="http://earthquake.linkeddata.it/resource/Device"/>
    <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A mechanism that controls the operation of a machine</ontology:description>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/ComputerSystem">
    <rdfs:subClassOf rdf:resource="http://earthquake.linkeddata.it/resource/ExperimentalComputationFacility"/>
    <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A system of one or more computers and associated software with common storage</ontology:description>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/ElectricalDamper">
    <rdfs:subClassOf rdf:resource="http://earthquake.linkeddata.it/resource/Damper"/>
    <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A damper that is commonly used as a ventilation </ontology:description>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/DepthGage">
    <rdfs:subClassOf rdf:resource="http://earthquake.linkeddata.it/resource/Sensor"/>
    <ontology:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A sensor which is a pressure gage that displays the equivalent depth in water.</ontology:description>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
```

Figure 5.9: A fragment of OWL ontology.

The equivalent class concept is important for queries because symbols can be folded together during search. The owl:equivalentClass can also be used to link ontol-

gogies. This sometimes may call a semantic join. The owl:equivalentClass property can be used with any class and its value must be an instance of class.

OWL, differentiate between properties that relate individuals to data values (datatype properties) and properties that hold between two individuals (object properties)(Antoniou and Harmelen , 2009). The owl:ObjectProperty and owl:DatatypeProperty are used to define the object and data properties. Furthermore, properties can also have sub-properties which are defined by the syntax rdfs:subPropertyOf. The rdfs:domain and rdfs:range syntax are used to classify the domain and range of properties, showing which class that a property is associated with and what type of values a property may have.

```
<owl:ObjectProperty rdf:about="http://earthquake.engineering.unitn.it/relation/used_in">
  <rdfs:range rdf:resource="http://earthquake.engineering.unitn.it/resource/Experiment"/>
  <rdfs:domain rdf:resource="http://earthquake.engineering.unitn.it/resource/Specimen"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://earthquake.engineering.unitn.it/relation/Generate">
  <rdfs:range rdf:resource="http://earthquake.engineering.unitn.it/resource/Specification"/>
  <rdfs:range rdf:resource="http://earthquake.engineering.unitn.it/resource/Signal"/>
  <rdfs:range rdf:resource="http://earthquake.engineering.unitn.it/resource/ComputerFile"/>
  <rdfs:domain rdf:resource="http://earthquake.engineering.unitn.it/resource/Experiment"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://earthquake.engineering.unitn.it/relation/Member_of">
  <rdfs:range rdf:resource="http://earthquake.engineering.unitn.it/resource/Project"/>
  <rdfs:domain rdf:resource="http://earthquake.engineering.unitn.it/resource/Person"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://earthquake.engineering.unitn.it/relation/has_device">
  <rdfs:range rdf:resource="http://earthquake.engineering.unitn.it/resource/Device"/>
  <rdfs:domain rdf:resource="http://earthquake.engineering.unitn.it/resource/Experiment"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://earthquake.engineering.unitn.it/relation/Performed_by">
  <rdfs:range rdf:resource="http://earthquake.engineering.unitn.it/resource/Person"/>
  <rdfs:domain rdf:resource="http://earthquake.engineering.unitn.it/resource/Experiment"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://earthquake.engineering.unitn.it/relation/Product_of">
  <rdfs:range rdf:resource="http://earthquake.engineering.unitn.it/resource/Sensor"/>
  <rdfs:domain rdf:resource="http://earthquake.engineering.unitn.it/resource/Signal"/>
</owl:ObjectProperty>
```

Figure 5.10: A fragment of OWL ontology.

In Figure 5.10, in the case of the property Generate we can state that it connects experiment with signal, Specification, ComputerFile. This information greatly helps in describing the internal structure of an ontology. OWL Lite class expression can only contain class names and property restrictions.

OWL 1 has been successful to design ontology but certain problems have been defined. Following we present the problems of OWL1 (Grau et al., 2009):

- Expressivity Limitations
- Qualified Cardinality Restrictions
- Relational Expressivity
- Datatype Expressivity
- Keys

OWL 2 introduces an XML syntax that presents several improvements for ontology web publishing. This syntax typically offers convenient and straightforward parsing and processing, equipped with XMLs wide adoption and tools support. OWL 2 is a major set of extensions and, mostly, improvements to OWL 1 which solve some of these problems. OWL had an abstract syntax to help with writing the specs but all OWL ontologies were expressed via RDF. Qualified cardinality problem was solved by using literal valued properties to identify resources; OWL2 introduce owl:haskey which provide a list of properties with both object and literal valued properties that identify resources of a given type.

Moreover, OWL 2 has three profiles, known also as fragments or sublanguages, which are independent of each other (Motik, 2012):

- OWL 2 EL can be used in applications which use ontologies with large number of properties and classes. The EL acronym refers that profile basis is in the EL family of DL that provide only Existential quantification;
- OWL 2 QL can be used where query answering is the most important reasoning task and in applications which use large volumes of instance data. The QL acronym refers to the fact that query answering can be implemented by rewriting queries into standard relational Query Language;
- OWL 2 RL can be used in applications requiring scalable reasoning without sacrificing too much expressive power. The RL acronym refers to the fact that reasoning can be implemented using a standard Rule Language.

5.4 Conclusion

We exploit the benefits which derived from a logic-based formalization of Earthquake engineering research projects management and experiments systems. Formalizing earthquake engineering terminology in RDF allowed to perform reasoning on the expressed semantic and consequently to evaluate the coherence of the mappings between them. After that, we also took advantage of the RDFS language for representing earthquake engineering terms and their inter-relations. Concerning this point, we could have used also OWL for the representation of our terminologies, but we would not have been able to take full advantage of its expressivity, so we have a simple structure composed mostly of a general hierarchical level (is-a relation and part-of) and few attributes assigned to each earthquake engineering term.

CHAPTER 6

LINKING RDF DATASETS

6.1 Introduction

The Semantic Web is a Web of Data, where related data are linked so that the machine can explore the web of data by crawling the links. This collection of interrelated data sets on the Web is usually referred to as Linked Data. Linked Open Data (LOD) is Linked Data which is released under an open license, and does not impede its reuse for free (Tim Berners-Lee , 2006). A five star rating schema for the linked open data is introduced by Tim Berners- Lee as follows:

- Data is available on the web with an open license.
- Data is available as machine-readable structured data (e.g. excel instead of image scan of a table).
- Data is available as (2), plus non-proprietary format (e.g. CSV instead of excel).
- All the above, plus use open standards from W3C (RDF and SPARQL) to identify things, so that people can link to it.
- All the above, plus link data to other peoples data to provide context.

The Semantic Web research community, and particularly the W3C Linking Open Data (LOD) project, aimed to bootstrap the Web of Data by identifying existing data sets available under open licenses, convert them to RDF according to the Linked Data principles and to publish them on the Web. As a point of principle, the project has

always been open to anyone who publishes data according to the Linked Data principles. The Linked open data cloud is illustrated in Figure 6.1.

Figure 6.1: Linked open Data Cloud (Tim Berners-Lee , 2006)

Many Semantic Web applications have been developed by accessing to the data sets in the LOD cloud, such as linked data browsers (Tummarello et al. , 2010), semantic search engines (Finin et al. , 2004), and some domain specific applications (Kobilarov et al. , 2009). Although many Semantic Web applications have been developed that demand access to the linked data sets, integrating ontology schemas or data sets from diverse domains remains a challenging problem (Bizer et al. , 2009). Furthermore, not all the ontology schemas are necessary for accessing to different data sets. For instance, when we want to link the ontology from a publication data set to a cross-domain data set, we only need to know the ontology schemas related to the publication in the cross-domain ontologies. Integrating heterogeneous ontologies can help linked data sets integration and missing links discovery. Additionally, integrating only essential parts of the ontologies and the alignments among various ontologies can improve the interoperability of the data sets and make it easier for the Semantic Web developers to understand how the instances are interlinked. Four fundamental challenges are introduced in (Auer and Lehmann , 2010) to feasibly establish the Web of Data:

- Improving the performance of large-scale RDF data management

- Increasing and easing the interlinking and fusion of information
- Improving the structure, semantic richness and quality of linked database.
- Adaptive user interfaces and interaction paradigms.

In this thesis, a tool for automatic linking of RDF datasets that consists of graph-based ontology integration is proposed. The developed system also retrieves core ontology schemas by applying string methods that can help Semantic Web application developers easily understand the ontology schemas of the data sets. Furthermore, the system enriches the integrated ontology by adding domain and range that can provide with rich information about the ontology. The integrated ontology can help us discover missing links, detect misused properties, recommend standard ontology schemas for the instances, and improve the information retrieval with simple SPARQL queries.

We discuss some background and sources such as ontology matching, analysis of sameAs links, and concept extraction. The matching algorithm is introduced, and ontology system architecture for the LOD cloud is discussed. The evaluation results are also reported in this chapter.

6.2 Linked data background and sources

Building explicit data linking systems aim to support end-users integrate and reuse their heterogeneous data. Hence, several attempts to assist in the creation of data linking have been presented in the last couple of years to overcome such heterogeneity, which supports to enhance interoperability between applications and/or systems. For instance, the UNIX pipes, gave a pathway that allows the user to chain the standard inputs and outputs of processes with one another, thus creating a pipeline of operations, where each operation relies on the data of another operation.

The most relevant related work to the proposed system are Yahoo Pipes[1] , IBM Mashup Center, SPARQLMotion[2], MashQL[3] and DERI pipes (Phouc et al. 2009).

[1] http://pipes.yahoo.com/pipes/

[2] http://www.topquadrant.com/technology/sparqlmotion/

[3] http://sina.birzeit.edu/mashql/

Mashup editor, for example Yahoo Pipes, allows people to write query inside a module and visualize these modules and their inputs and outputs as boxes connected with lines. Yahoo Pipes is in general a powerful environment for databased mashup. However, Linked data source is not sufficiently supported; there is no direct module for including SPARQL endpoints even more to request RDF syntaxes via HTTP content negotiation. Considering the point we can say Yahoo Pipes is currently not suited for integrating content from the semantic web and it breaks the idea of Linked data.

Recent approach in the semantic web community is DERI Pipes[4] inspired by Yahoo's Pipes, which is an engine and graphical environment for general Web Data transformations and Mashup supports RDF, XML, Micro formats, JSON and binary streams. It will be used as a "Web Pipe" embedded in the applications which will work as a mashup command Line tool supports SPARQL, XQUERY and several scripting languages. It will be extended as needed DERI Pipes, and will produce an output streams of data (e.g. XML, RDF, JSON) that can be used by applications. However, when invoked by a normal browser, they provide an end user interface for the user to enter parameter values and browse the results. While DERI Pipes provides substantial support for applying the idea of piping to the RDF world, it is not properly linked with popular RSS and Atom feed environments. There are also tools which let users to build a custom analyzer as DERI pipe does. The MashQL facilitates users to query and mashup massive amount of structured data on the web intuitively. Open data Mashup[5] also provides similar functionalities like DERI pipe; it offers visualization based on vocabularies and also supports map visualization. Furthermore, we found a SILK framework [6], a tool that provides support interlinking between entities within different Web data sources. The SILK framework introduces Silk linking specification language (Silk-LSL) that allows the users to write scripts for specifying conditions to build interlinking (Volz et al., 2009); whenever resources are matched with a given threshold , the tool outputs sameas links.

[4] http://pipes.deri.org/

[5] http://ogd.ifs.tuwien.ac.at/mashup/

[6] http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/

6.3 Ontologies alignment using Linked Data

The ontology heterogeneity problem in the LOD cloud induces the difficulty of accessing to various data sets and remains as one of the most challenging problems in the Semantic Web research. Ontology integration is defined as a process that generates a single ontology from different existing ontologies. Ontology alignment or ontology matching is commonly used to find correspondences between ontologies to solve the ontology heterogeneity problem (Pavel and Euzenat , 2013). DBpedia is a source of structured information extracted from Wikipedia containing about 1.5 million objects that are classified with a consistent ontology. Because of the diversity of the data in DBpedia, it presents itself as a hub for links in the Web of Linked Data from other sources (Auer et al. , 2007). As DBpedia contains a large variety of data (e.g. abstracts, links to other articles, images, etc.), we limit our approach to RDF containing the rdf:type assertion and info boxes, which provide factual information.

SPARQL Protocol and RDF Query Language (SPARQL) are powerful RDF query languages that enable Semantic Web users to access to the Linked Data (Heath and Bizer , 2011). However, the users have to understand the ontology schemas of the data sets in order to construct SPARQL queries. Querying with a simple ontology that integrates various ontologies can simplify SPARQL queries and help Semantic Web application developers easily understand the ontology schemas so that they can retrieve rich information from various linked data sets. Another problem of dealing with the LOD cloud is that not all the data sets in the LOD are trustworthy. For instance, the data publishers sometimes make mistakes when they convert data into RDF triples. They may use different terms of the properties for the same concept. For example, static_test" is represented using StaticTest", statictest" and so on. Furthermore, some of the instances are described with general ontology classes rather than specific classes. These mistaken data should be corrected, but it is time-consuming and infeasible to manually inspect large ontologies of the linked data sets to discover these mistakes.

Moreover, instance may be noisy, if none of the triples of the instance contains information that can represent the characteristics of the instance. For example, if all the triples of an instance are sameAs links or broken links, we cannot learn any

information that can represent the characteristics of an instance. We remove these noisy instances from the core data set before collecting predicates and objects using sparql filter parameter. Experiments show that the graph-based ontology integration can understand the characteristics of the interlinked instances at both class and property levels. By combining related classes and properties from various data sets, we can find sameAs links and reduce the ontology heterogeneity problem that help Semantic Web application developers easily understand the relations between different ontologies without any manual inspection.

We combined the String-based and Knowledge-based ontology matching methods on the predicates and objects to discover similar concepts. Moreover, identifying common instances between the two ontologies required for this technique using the owl:sameAs links, where the instance identifier in each ontology gets replaced with a combination of the URIs from both ontologies. In the alignment process, instead of focusing only on classes defined by rdf:type which we will call restriction classes that help us identify existing as well as derived set of classes in an ontology. A restriction class with only a single constraint on the rdf:type property gives us a class already exist in the ontology, for example in SEPREMO the restriction identifies the class Device.

Our aim is to automatically construct a simple ontology that integrates ontology schemas from various linked data sets. By collecting linked instances, we can identify different concepts that indicate identical or related information.

6.4 Ontologies alignment using Linked Data

In this section, we give overviews of the implementation of the semantic web matcher including algorithms and system design; the proposed approach elaborates how to construct dynamic semantic data linking by taking advantage of DERI pipe (Dunne et al., 2011) features.

6.4.1 Algorithm

The implementation of DERI pipe includes an online AJAX pipe editor, and execution engine. Moreover, users cannot manage user interface of pipe without having to

know specific syntax even more SPARQL language that is used to configure some of the functional blocks. To address the issue and overcome the limitations of current approaches, the step by step implementation of the semantic matcher is illustrated in Figure 6.2 and described below.

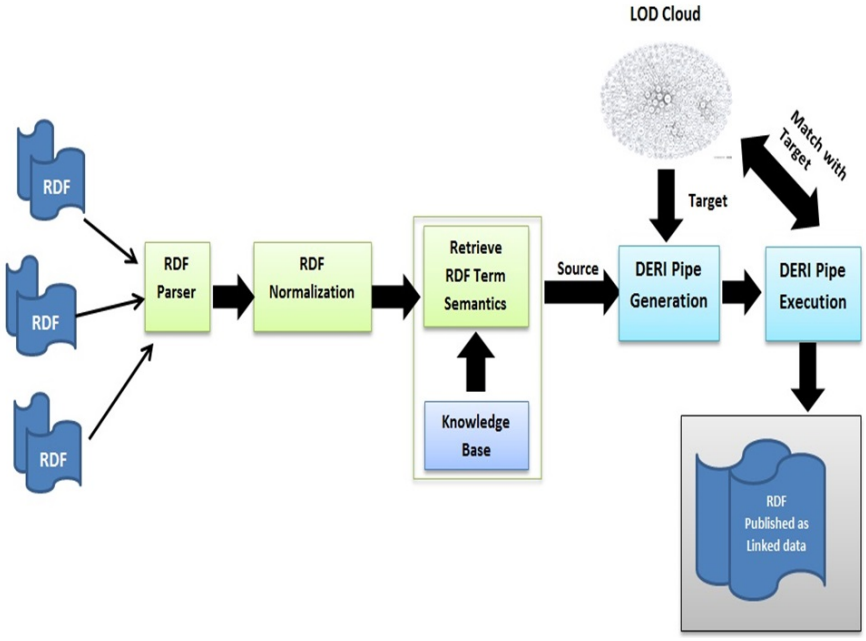


Figure 6.2: The basic structure of a Semantic matcher algorithm.

1. Select source datasets published in RDF. During select resources in the target dataset that can match a specific source resource. We first select the labels that represent the source resources. Therefore, RDF parser objects that reads the InputStream and creates RDF statement out of it. It is then stored in the memory.

2. RDF Normalization. Due to the various adopted RDF data formats heterogeneity problem raise; we need to facilitate users to mashup without any knowledge of the underlying heterogeneous data sources. Therefore, all namespaces have to be fixed in the semantic matcher using the normalize method that performs a transformation on the input that results in all aspects of the graph being arranged in a deterministic way. We consider for every entry in the node begins with a string (last part of namespace). That is why the following scenario in the developed matcher are performed.

- If the last part of the namespace prefix is numeric value, replace with rdfs:label if only if rdfs:label contain string value.
- If the nodes contain #, %, spaces and dot, replace with underscore.

3. Retrieve RDF Term From Knowledge Base. After normalizing RDF we retrieve all terms from Knowledgebase for example WordNet. For our experiment, we consider only queries each triple and examine one component i.e., the subject of the each triple.

4. Select target datasets published in RDF. We use the Linked open data cloud (e.g., DBpedia) as a target datasets to match between terms. We get all terms from the user selected input file and put concepts as a string in the last part of the target URI. More precisely, entity datasets of the source is used to put entities in the target dataset.

5. DERI Pipe Generation. Using DERI pipe mashup frameworks perform entity matching strategies that include an equivalent relation. We first get every single node from the user selected files as source resource and set DBpedia as a target in the Pipe. Then looping over nodes and pass the value as a string in the Pipe which is performed by using C-operator (Morbidoni et al., 2007).

6. Performing matching. After creating the Pipe, pass Pipe (written in simple XML syntaxes) as a string in the DERI pipe execution Engine. When string invoked in the execution engine, the engine fetches data from remote sources into an in memory triple store, and then executes the tree of operators. Finally, it uses filter clause that allows refines the output.

7. Output generation. If the process of matching succeeds, we get sameAs relations. When the engine is executed, HTTP caching is performed to avoid recomputing a pipe output if the sources remain unchanged.

This abovementioned approach offers several advantages:

- The link specification is simplified, reducing the manual input;
- The alignment can be reused for linking any two datasets described according

to these two ontologies;

- There is a clear separation between links, linking specification, and ontology alignments.

6.4.2 System Design

Figure 6.3 describes the overall web based system architecture, providing detailed information about the system.

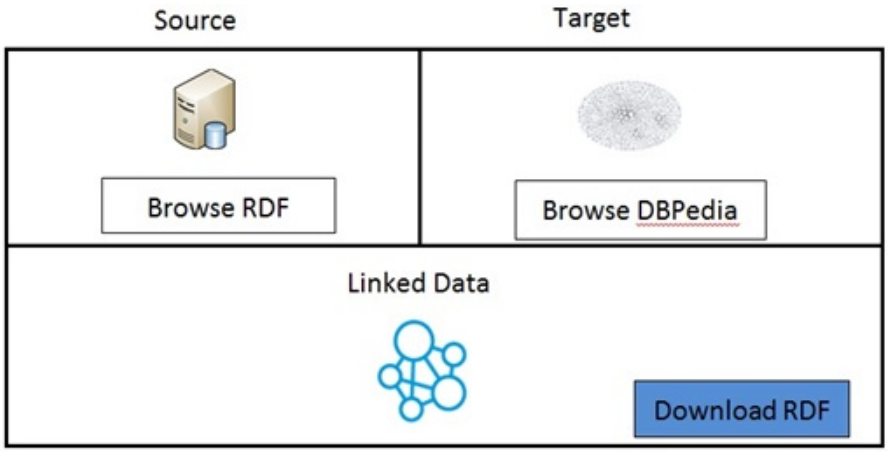


Figure 6.3: System Architecture

At the first step, users are able to select source - when the upload button is clicked, the selected RDF file from the local system is uploaded to the Semantic Web Matcher Server and the file is added to the server file system. The next step is to set the target datasets (e.g DBpedia) - currently our system lacks functionality to select target data sets. Once a user clicks the match button, DERI Pipes execution engine runs and gets the direct matches between the source and target datasets with the similarity relationships. Finally users can download the output is an HTTP-retrievable RDF model.

6.5 Evaluation

This section discusses the evaluation process of the proposed approach using RDF data sets, e.g., WordNet Hyponym relation, SEPREMO, which exist in the web. We

have also evaluated the computational performance to execute the RDF files. For this experiment, we used two different RDF source files, i.e., a data source with length 15 KB and 563 KB content length.

Matcher is a one-time process that loops over entities or concepts repeatedly. To evaluate the performance of the semantic matcher, two different tests were carried out, each test and its values were calculated, which are presented in Table 1. These results indicate the matching rate and time to execute the RDF files.

Source	Source Entity Amount	File Size	Target	Found Matches	Correct	Execution Time
WordNet Hyponym	100	15 KB	DBPedia	100	100%	2 minutes
SEPREMO	57	23 KB	DBPedia	16	28%	1 minute

Table 6.1: Performance of Semantic matcher

Moreover, the result shows that now Semantic matcher can find some of the correct sameAs relation. However, the SEPREMO ontology result is not as good as expected. This is because most of the terms in that ontology are currently not available in the DBpedia.

6.6 Conclusion

This chapter presented the automatic RDF datasets linking algorithms, including a detailed discussion of the calculations used to determine similarity between two entities from different ontologies. We have also shown that the algorithm converges to a solution. The experimental results presented illustrate that tools outperforms most existing ontology matching algorithms, and obtains accuracy values. We have also shown that the process of semantic verification enhances the performance of the system. The system is designed for integrating heterogeneous LOD ontologies. The developed system consists of graph-based ontology integration and solves main problems. The graph-based ontology integration solves the ontology heterogeneity problem that is one of the most challenging problems in dealing with the Linked Open Data.

Experimental results show that core classes and properties in each data set is discovered, which can help data publishers detect misuses of ontologies in their pub-

lished data sets. The method is domain-independent and can be performed on data sets from various domains. In addition, for the instances of a specific class, we can recommend core properties that are frequently used for the instance description. Although we need minor manual revision on the automatically created integrated ontology, the ontology integration method successfully retrieves related ontology classes and properties that are critical for interlinking related instances.

CHAPTER 7

INTELLIGENT QUERY ANSWERING

7.1 Introduction

Recently, information retrieval has been a challenging research issue because of the huge development in information resources and technology. One of the most successful approaches in information retrieval systems is to annotate the file to give additional descriptions of the archived information. To achieve the aim of the Semantic Web, the resources, seismic engineering text or multi-media must be semantically tagged by metadata so that heterogeneous applications can exploit them. There are many techniques for the semantic annotation of documents, despite their growing number complexity and potential impact on retrieval. However fully automated intelligent query systems face a number of challenges that are not easy to tackle, for example:

- No mechanism exists, which can efficiently retrieve a required file
- The lack of sophisticated semantic, syntactic and conceptual processing to generate answers

The traditional techniques used for document retrieval systems include stop lists, word stems, and frequency. The words that are deemed irrelevant to any query are eliminated from searching. The words that share a common word stem are replaced by the stem word. The occurrence here can be simply the frequency of a word or the ratio of word frequency with respect to the size of a document.

This chapter consists of four subsections, beginning with document collection which provides a description of document collection procedure. The following subsection discusses about Indexing and searching and presents detailed overview of the system design. Then in the last section we present system overview and finally conclude the chapter.

7.2 Document Collection

The document collection was composition of Seismic Engineering documents from Internet and documents provided by our research groups. The files provided for the shared task were available in PDF, Text and DOC formats, this collection were needed to be processed need to be processed before querying, in order to transform them into a form which is appropriate for topic answering. Suppose that a query is posed to find all documents that describe Device# and Device. This type of queries cannot easily be processed in relational document databases or object-oriented document databases due to inflexible modeling of irregularity of documents and unacceptable performance. Using Apache Lucene we can perform this type of query. To measure the quality of query matching, our empirical datasets do not only have to contain an appropriate number of real-world queries, but must also contain details about the similarity between these queries. Nevertheless, the document is suitable to show the quality and performance of our combined query matching approach compared to the methods. A document may have more than one field with the same name added to it. All of the fields with a given name will be searchable under that name.

7.3 Indexing and Searching

7.3.1 Apache Lucene

Apache Lucene[1] is an open-source, high-performance, full-featured text search engine library written entirely in Java. Search engines deal with the measurement of how close the source information matches with the user input; thus retrieving the

[1] http://lucene.apache.org/

most relevant information to the users. In order to calculate the relevancy, search engines use several parameters such as the popularity of a document, the date of a document, user preferences. Lucene has become exceptionally popular and is now the most widely used information retrieval library extracted from usage logs.

Lucene has a directory named index, which contains files used by Lucene to associate terms with documents. To accomplish this, a Lucene index was created with a specific analyzer model-dependent. An Analyzer takes a series of terms or tokens and creates the terms to be indexed. A unique kind of Lucene index has been used for all developed models, or in other words, all models share the same Lucene index. Lucene is full-featured and provides

- Speed: sub-second query performance for most queries
- Strong out of the box relevancy ranking.
- Complete query capabilities: keyword, Boolean and +/- queries, proximity operators, wildcards, fielded searching, term/field/document weights, find-similar, spell-checking, multi-lingual search and more
- Full results processing, including sorting by relevancy, date or any field, dynamic summaries and hit highlighting
- Portability: runs on any platform supporting Java, and indexes are portable across platforms
- Scalability: small RAM requirements and incremental indexing as fast as batch indexing.
- Low overhead indexes and rapid incremental indexing.

Documents and fields are Lucene's fundamental units of indexing and searching. It is a container that holds one or more fields, which in turn contain the real content. Each field has a name to identify it, a text or binary value, and a series of detailed options that describe what Lucene should do with the field value when we add the document to the index. To index our collection sources, we must first translate it into Lucene's documents and fields. It allows duplicate fields to be added to a Document.

This can make updating documents easy because it allows adding fields of various reports into a single Lucene document, with a mixing of all visit reports. At the end, we have the visit reports and the fields of various reports sharing the same Lucene field.

Queries are formal statements used for requesting information from search engines. Search engines analyze queries and reply with the most relevant document list. Retrieving documents is the next part of a search engine. Search engines select relevant documents from a document collection. In order to retrieve relevant documents, indexers assign scores to documents by using various parameters such as zone, date, page rank. In addition, search engines may assign additional scores with respect to the queries. After the scoring step is completed, search engines order result set with respect to their scores. In information retrieval, ordering the results of a query by using various scores is called ranking. Figure 7.1 represents Apache Lucene core architecture overview:

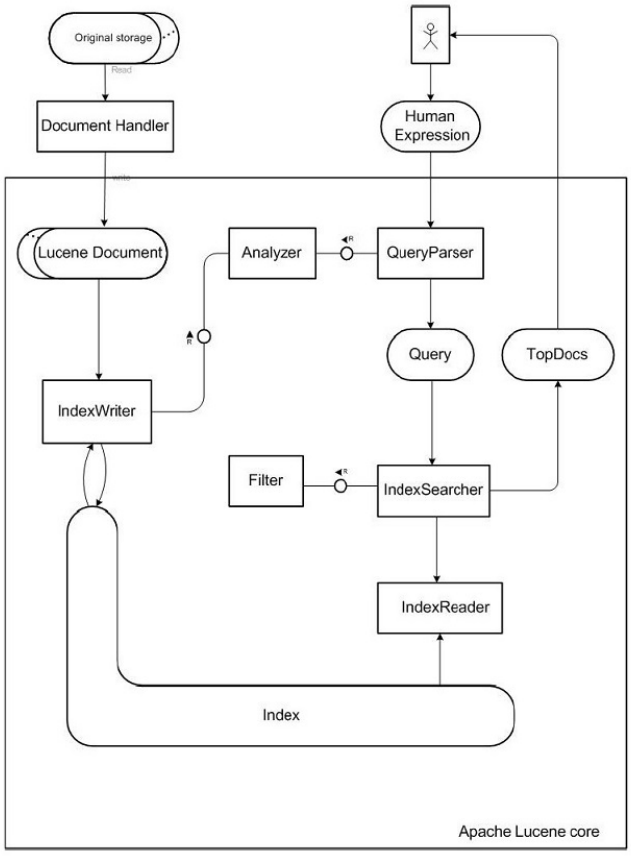


Figure 7.1: Apache Lucene System Architecture

Apache Lucene with Annotated document

Document annotation and search have received tremendous attention by Earthquake Engineering and Semantic web communities (Handschuh and Staab , 2003).Annotations help users to easily organize their documents. Also, they can help in providing better search facilities: users can search for information not only using keywords, but also using well-defined general concepts that describe the domain of their information need. Although traditional Information Retrieval (IR) techniques are well-established, they are not effective when problems of concept ambiguity or synonymity appear. We have designed and implemented an easy-to-use document annotation framework that supports the most widely used document formats, also providing ad-

vanced search facilities.

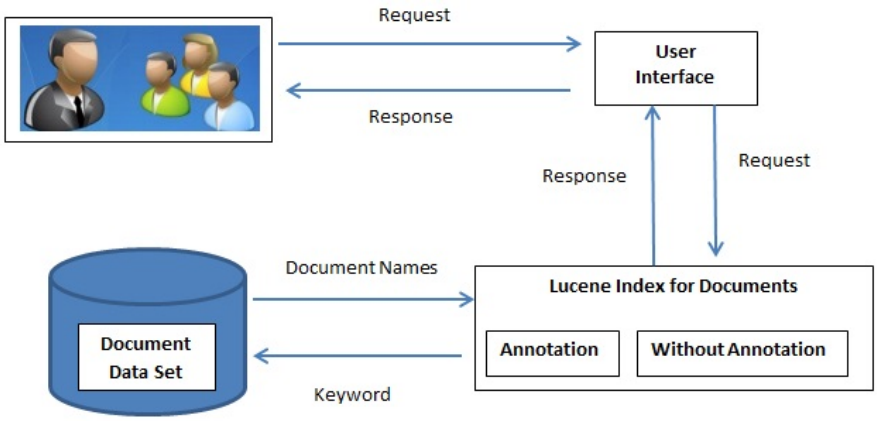


Figure 7.2: Intelligent query answering system architecture

The framework is based on a server-based architecture, where documents with annotations and without annotations are stored in a central. This offers a collaborative environment where users can annotate and search documents. After providing search words then search method first checks if Lucenes index already exits. If so, it searches on the existing index. If not, the search method first calls the method provided by Lucene Index to create the index, and then it searches on the newly created index. After the search result is returned, this method fetches the needed attribute from the search results and generates an instance for each search result. At last, the instances are put into a list and returned to the Request Manager subsystem.

Our developed system contains two private fields: data directory and index directory. Data directory represents the directory that stores all the files to be indexed, and index directory represents the directory used to store the Lucene index. The Index Manager class provides two methods: create Index and add Document. We use create Index to create the Lucene index if it does not exist, and then use add Document to add one document to the index. In this scenario, one document is a file. This method calls the methods provided by the Document Parser class to parse the file content. This class extracts the text content from the file. We provide three methods in this class: get content, get title, and get path. The first method returns the file contents without tags, the second method returns the title of the file, and the last

method make the path of the file.

We categorize the basic search facilities of our framework into Keyword-based search; this is the traditional search model. The user provides keywords and the system retrieves relevant documents based on textual similarity. We adopted the text similarity metric used in Lucene engine. Keyword-based search returns an ordered Result Set of tuples that contain all the documents matched with the terms. The similarity score based on document textual similarity with the searching terms.

Our proposed approach to analyze annotations of documents improves document relevance estimation during the search in materials. Hence, technology had been developed to make the contents and special properties of document accessible and searchable. We described key issues encountered during the developments user interface of our search engine which takes into account the special characteristics of documents during the indexing process.

7.3.2 Apache Solr

Apache Solr[2]is the most popular, fast, scalable, open source search engine built on Apache Lucene Project. Solr is standalone full-featured search engine that provide all capabilities that we need to index and retrieve documents. Since Solr is a wrapper of Lucene library, it provides all capabilities of Lucene. We use Solr in order to index document (labeled or not labeled) and retrieve them. Solr is a standalone enterprise search server with a REST-like API (Smiley and Pugh , 2011) and Lucene is a high performance and scalable IR library (McCandless et al. , 2010). Figure 7.3 presents Solr system architecture:

[2]http://lucene.apache.org/solr/

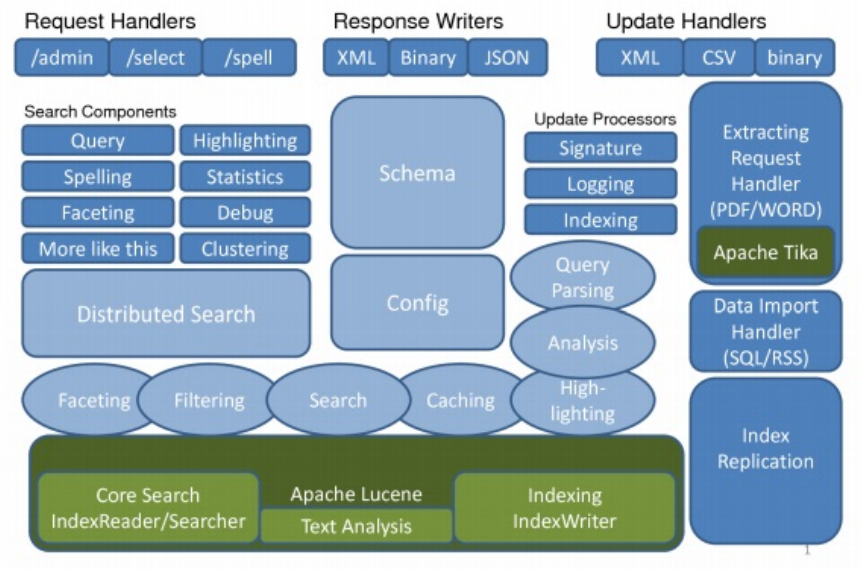


Figure 7.3: The architecture of Solr (Yonik , 2006)

A Solr index stores a set of objects, each consisting a list of possibly replicated and unordered fields associated to a value. Each object is referable by a unique identifier generated by the index at indexing time. The query can be done on any of the input tags as specified while indexing (id, text, title and so on). The query is of the form *:* where the first * represents the field on which the query has to be done and the second * represents the keywords for which the documents has to be searched for. The tool is quite efficient in searching data based on the related queries. The raw data is first provided to the tool in the form of text or xml files which gets indexed and stored inside it. The indexing is completely a property of Solr which can be controlled by modifying the internal schema of Solr. Once indexing is completed, querying can be done by providing the keywords as query. The tool also has a unique feature of boolean queries like AND and OR. When a collection of keywords are provided as search query to Solr with Boolean AND in between the keywords, then we get the intersection of the documents which contain the keywords. This brings down the number of searched documents which also helps in processing. Admin panel of Apache Solr given below(Figure 7.4):

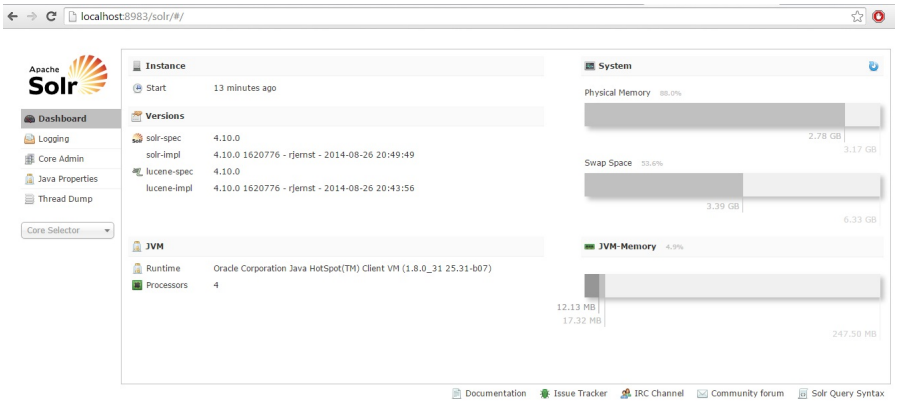


Figure 7.4: Figure: Solr Admin Panel

The Solr admin panel has following benefits:

- load pages quicker
- access and control functionality from the Dashboard
- reuse the same servlets that access Solr-related data from an external interface, and
- ignore any differences between working with one or multiple cores.

When data is added to Solr, it goes through a series of transformations before being added to the index. This is called the analysis phase. Examples of transformations include lower-casing, removing word stems and so on. The end results of the analysis are a series of tokens which are then added to the index. Tokens, not the original text, are what are searched when user perform a search query. Indexed fields are fields which undergo an analysis phase, and are added to the index. Displaying search results to users, they generally expect to see the original document, not the machine-processed tokens. The purpose of the stored attribute: to Solr to store the original text in the index somewhere.

7.4 Output

The interface of developed search engine is shown in Chapter 8. This platform developed with using Apache Lucene and users can search documents by keywords. Each retrieved document is shown in a page, with a link to its original file name. The hyperlink of each document links users to a page showing all the documents.

7.5 Conclusion

Traditional search engines do not provide a reasonable way to manage domain specific documents and information on these documents cannot be fully extracted and document relations are not present to users. In order to solve this problem, we propose a domain specific search engine that is capable of parsing user queries to intercept the usage of annotation kept in memory and, in this case, to manipulate the query response to deliver the set of documents. Developed system is suitable for further extension for example, a search engine integrated with ontology metadata and relation between them.

CHAPTER 8

RESULT

8.1 Introduction

In this chapter, we describe basically what advantages users can get with KB-based systems over traditional DB systems. SEPREMO is a web based platform generally used to visualize a class hierarchy of entities includes its instances and metadata. The SEPREMO user interface divided into three parts; the first part represents hierarchy of concepts that can describe the characteristics of the entity for example cardinality of the relationship like synonym and transitive or more specific. Moreover, user queries can be formulated using SPARQL. In the second part we represent instances that map with each entity. We consider the entire instance as the textual content which store in the local repository system. For example, Experiment has list of experiment name which conducted in the laboratory; here name of the experiment consider as instances. Finally, each instances contains four metadata set includes report, graph, RDF and excel file of the experiment entity whose information can also be stored in file based system.

Moreover, due to lack of data integration among seismic laboratories belonging to the RELUIS network, there is an urgent need of creating a unique platform for Italian Universities Laboratories capable of sharing seismic experimental data and knowledge. Therefore, a central database where centralized access to database nodes that are distributed over the network is needed. This database will be able to dialog with a central portal in a uniform manner. According to the same perspective, and in order to foster a sustainable culture of co-operation among all of the Italian

research infrastructures and teams that are active in seismic experimental activities, the implementation of a distributed hybrid simulation framework was set.

The rest of the chapter is organized as follows. In subsection 8.2.1 depicts an ontology based information management system development approach. Subsection 8.2.2 describes experimental data collection procedure. While Subsection 8.2.3 demonstrates the architecture of the final system that was built on top of the integrated ontology, Subsection 8.2.4 reports evaluation results that show the effectiveness of the ontology. On the other hand, in section 8.3 explains the RELUIS database from the perspective of external users and how they can take advantage of this RELUIS infrastructure.In Subsection 8.4 we conclude the chapter

8.2 Ontology Development

8.2.1 Approach

Figure 8.1 describes an ontology based information management system development approach that involves standard three-tier architecture. KB works as a backend of the system hosting ontologies represented in RDF, while query processing, inference mechanism and reasoning are incorporated in the business logic layer. Issuing queries and showing the corresponding results are supported by the User Interface (presentation) layer. However, for ontology development we follow the DERA methodology (Giunchiglia and Dutta, 2011), for ontology representation in RDF we use Jena and for ontology integration we implemented a facet based algorithm.

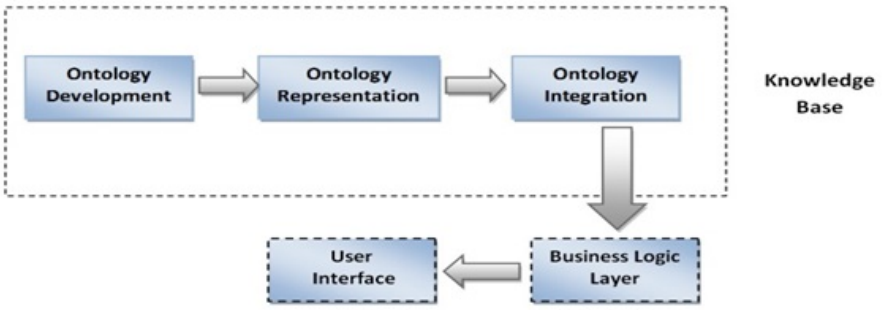


Figure 8.1: Ontology based development Approach

KB works as a backend of the system hosting ontologies represented in RDF while query processing, reasoning and inference mechanism are incorporated in the business logic layer. The operation of this layer is implemented with the use of the Virtuoso JENA API that serves also as a means of communication with the knowledge base layer. The business logic layer is also responsible for the ontological data loading, coming through the front end and based on the ontological schema of the back end. User queries and corresponding results are shown in the User Interface (presentation) layer. The presentation tier was implemented using HTML, CSS (Cascading Style Sheets) and XML (Extended Mark-up Language) to easily interoperate with other applications. So that the client can work more rapidly and thus ease the server of the burden of user validations, technologies like JavaScript and Java are used at this level.

8.2.2 Experimental Data Collection

In this subsection, an experimental test on a piping system under earthquake loading carried out by (Reza et al. , 2013) is briefly discussed to provide the reader with an overview of experimental data acquisition (DAQ) procedure.

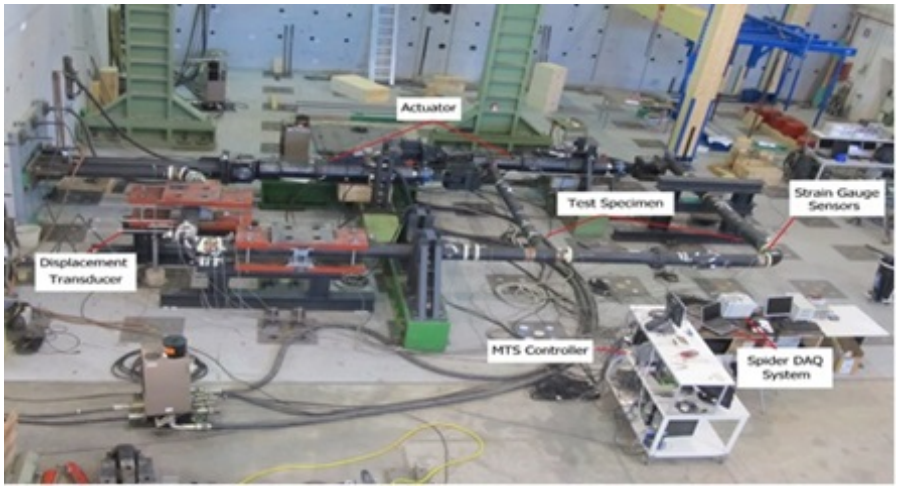


Figure 8.2: Experimental set-up of a piping system tested under earthquake loading (Reza et al. , 2013).

Figure 8.2 illustrates the relevant set-up of the experiment. As can be seen in this figure, the test specimen, i.e. the piping system, is excited with earthquake loading by means of two actuators which are controlled via an MTS controller. The test specimen is mounted with several sensors, such as strain gauges and displacement transducers, in order to observe its responses under applied seismic loading. In this particular experiment, four Spider8 DAQ systems were used to collect data from the sensors. Generally, output from a sensor, e.g. displacement transducer, is found in voltage, which is then transformed in another unit, such as mm, through a predefined calibration made in the DAQ measurement software. This data are then stored in a computer in an easily manageable format, such as Matlab (.mat) excel or ASCII, which are published in the ontology.

8.2.3 Experimental Setup

In Figure 8.3, we describe the process of creating the KB. The domain specific ontology that we developed was published into RDF by means of Jena (a Semantic Web tool for publishing and managing ontologies) and integrated with WordNet RDF using the approach described in Section VI. In order to increase the coverage of the background knowledge in the KB, we performed the integration of the two ontologies. The outcome of the ontology integration was put in Virtuoso triple store.

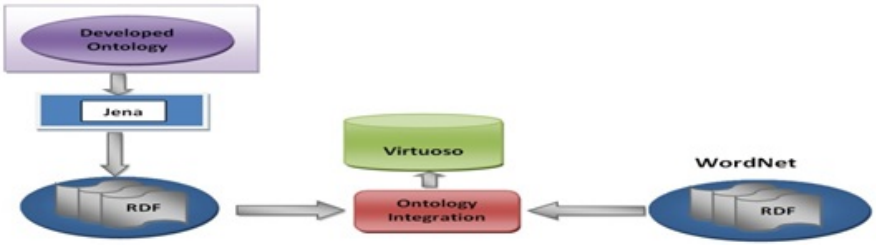


Figure 8.3: Ontology Integration and Population to KB

Figure 8.4 illustrates the architecture of our KB-based information management system that uses Semantic Web tools and technologies. As presented in the figure, the system is organized into three layers, which are User Interface (UI), Middleware

and KB.

To execute any user request, for example, visualizing the whole ontology or part of it, the corresponding service is called from the middleware. Each service communicates with the KB using SPARQL query. SPARQL is a query language especially designed to query RDF representations. It allows add, update and delete of RDF data.

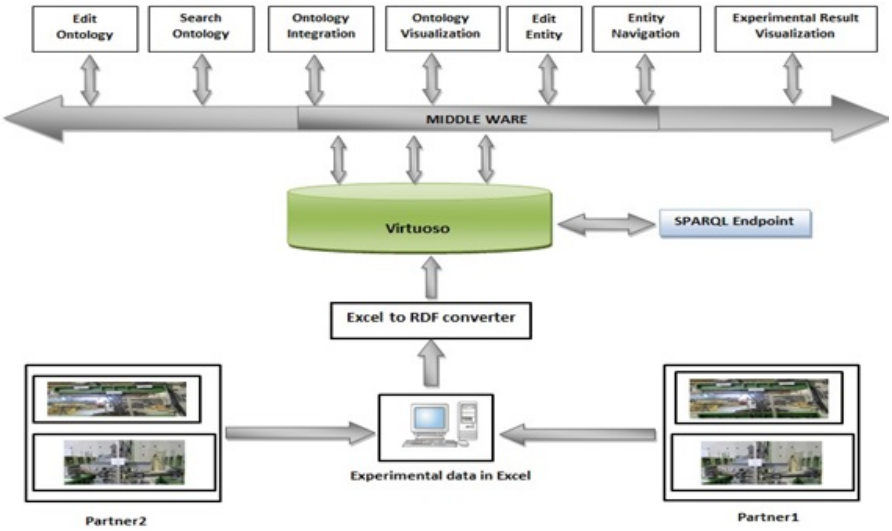


Figure 8.4: KB-based System Architecture

User Interface: Developed user interface allows people to perform the following operations on the ontological TBoxes: edit, search, integration and visualization, which are shown in the upper-most layer of Figure 8.4 alongside the following operations defined to be performed on the ABoxes: edit entity, entity navigation and experimental result visualization. With the edit ontology operation, concepts and relations can be created, deleted and updated. With the search ontology operation, concepts can be queried with their natural language labels. For the aggregation of an external ontology with the ones already present in the KB we perform the integration operation. In order to view and surf any of the ontologies, we employ (ontology) visualization operation. Note that in the KB until now we have two ontologies, WordNet and SEPREMO.

Edit entity operation is designed to help perform create, delete and update entities. Existing entities can be viewed and browsed with the entity navigation operation and experimental results can be shown with the corresponding visualization operation.

Middleware: All the functionalities germane to the operations that can be requested and eventually be performed from the user interface are implemented as services and deployed on a web server. Each service is basically communicating with the KB to execute one or more of the CRUD (create, read, update and delete) operations on its knowledge objects.

KB: This is our Knowledge Base hosting the ontologies consists of concepts and relations thereof, entities and their attributes and relations, and exogenous data from our own experimental setup and the one of our partner university, the University of Napoli.

In Table 8.1, we report the detailed statistics about SEPREMO ontology. This ontology consists of 11 facets, 193 entity classes, 6 relations and 13 attributes. Note that each of the entity classes, relations and attributes represents an atomic concept. Hence, in total we found 212 atomic concepts in the ontology and out of them 100 concepts are available in WordNet.

Object	**Quantity**
Facet	11
Entity class	193
Relation	6
Attribute	13
Concept	212
Concepts found in WordNet	100

Table 8.1: Statistics about SEPREMO ontology

8.2.4 Controlled Experiment

In this section we include different type of experiments carried out test methodological guidelines proposed in this thesis.

Synonym Search

When a concept is represented with two or more terms, they are essentially synonymous and can be represented in RDF with ***owl:equivalentClass***. For example, test and experiment represent the same concept and in the ontology they are encoded accordingly with equivalent relation. Therefore, as can be seen in Figure 8.5, user query for test can also return experiment because they are semantically equivalent.

Figure 8.5: Synonymus relationship of Test.

More specific concept search

In our ontology concept hierarchies are represented using ***rdfs:subClassOf***. For example, hammer and damper are more specific concepts of device, hence, they are represented as follows: hammer ***rdfs:subClassOf*** device; and damper ***rdfs:subClassOf*** device.

Figure 8.6: Transitive Relationship of Device

Moreover, hydraulic damper is more specific than damper and it is encoded as hydraulic damper rdfs:subClassOf damper. Note that rdfs:subClassOf is a transitive relation. Using OWL inference engine, we can utilize the power of transitivity and for a given concept we can retrieve all the more specific concepts that are directly or indirectly connected by rdfs:subClassOf. Therefore, a search for device retrieved all of its more specific concepts as shown in Figure 8.6.

Sparql Endpoint

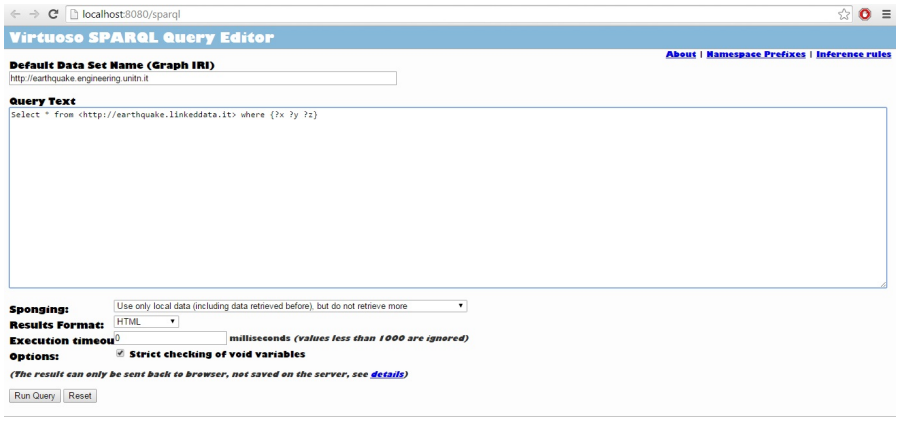


Figure 8.7: Sparql Endpoint

Figure 8.7 presents Sparql endpoint over the SEPREMO data set. The endpoint is provided using OpenLink Virtuoso as both the back-end database engine and the HTTP/SPARQL server. The endpoints usually support different result formats:

i. XML, JSON and plain text (for ASK and SELECT queries)
ii. RDF/MXL, NTriples, Turtle and N3(for DESCRIBE and CONSTRUCT queries)

A SPARQL endpoint enables users (human or other) to query a knowledge base via the SPARQL language. Results are typically returned in one or more machine processable formats. Therefore, a SPARQL endpoint is mostly conceived as a machine-friendly interface towards a knowledge base.

Ontology Visualization

Figure 8.8 presents ontology visualization features that use circle pack layout to present an entire overview of the whole knowledge base. Concepts are displayed as circles and sub-concepts are presented inside their parents; to increase the readability and to avoid confusion only the labels of parent concepts are displayed. The user can zoom by clicking on circles to display the parent concepts including child concepts. Circle Pack visualization provides a useful alternative by representing hierarchical relations through containment. It is possible to see an overview of the overall structure and the position of a certain concept.

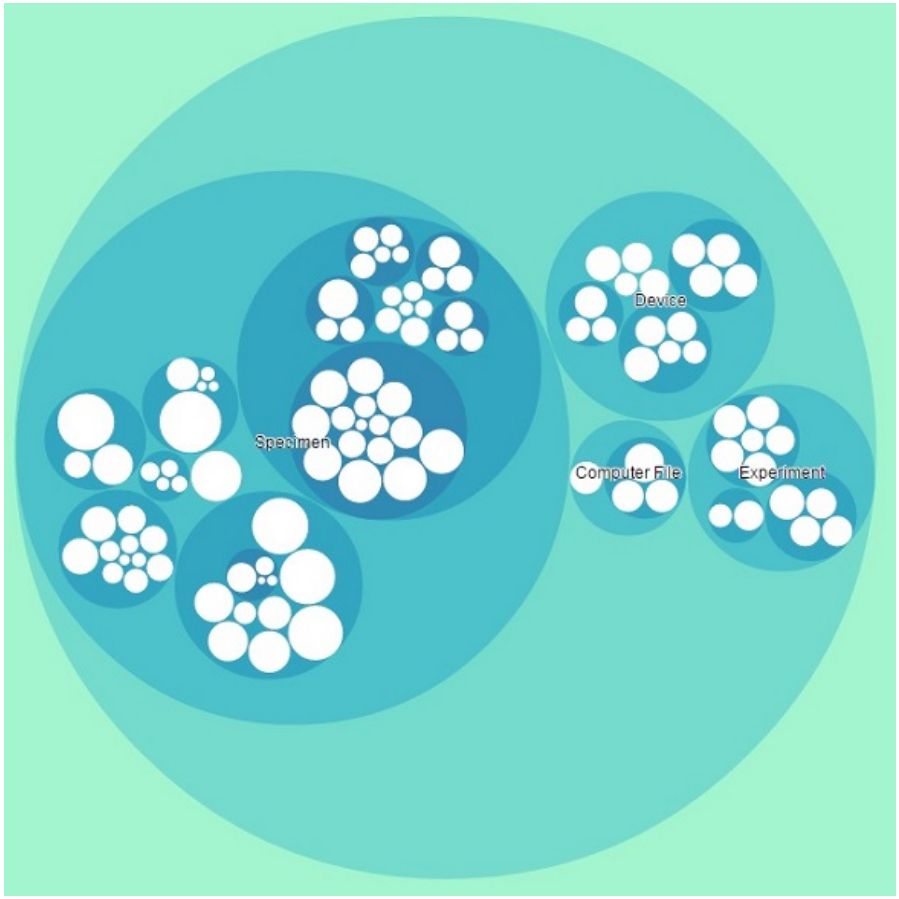


Figure 8.8: Circle pack layout for ontology visualization

Experimental Data Visualization

Information visualization can play an important role in the iterative model refinement process. In this domain, visualization is typically used for hypothesis generation, not hypothesis verification. Visually displaying the gathered quantitative measurements in the context of the graph model supports the hypothesis discovery process by allowing researchers to spot trends. Seismic Engineer use interaction graphs to model the behavior of experimental systems. Experimental data visualization which extracts the information clearly and effectively through graphical means that can be communicated others easily. These graphs serve as a form of dynamic knowledge

representation of the seismic engineering system being studied and evolve as new insight is gained from the experimental data. Using this vocabulary, some experimental datasets have been published as RDF data (Figure 8.9), which are then linked with DBPedia, the nucleus of the LOD Cloud.

```
<rdf:Description rdf:about="http://earthquake.engineering.unitn.it/resource/Experiment1">
      <rdf:type rdf:resource="http://earthquake.engineering.unitn.it/resource/Experiment"/>
      <earthquakeontology:load_displacement>(0.620.52, 338.81 61.28, 375.41 122.02, 411.45 182.77)
      </earthquakeontology:load_displacement>
</rdf:Description>
```

Figure 8.9: A snippet of the experimental data represented in RDF

Developed application that can assist in understanding the characteristics (exemplified in Figure 8.10) of the various experiments modelled in the SEPREMO ontology, we leveraged the generated RDF datasets.

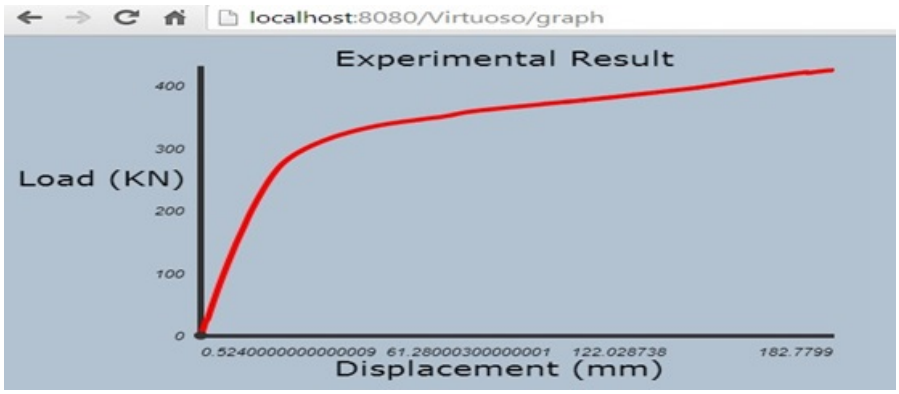


Figure 8.10: Load-displacement characteristics curve drawn with the data provided in Figure 8.9

User query for specific experiment and get all information related experiment including metadata; metadata includes experiment reports, experiment result in RDF format and Graph format to understand the experiment characteristic.

8.2.5 RDF Liking Datasets

A fundamental essential of the Semantic Web is the existence of large amounts of meaningfully interlinked RDF data on the Web. In the context of Linked Data, the problem of instance matching can be defined as follows: given two distinct RDF datasets A and B, find pairs of resources, one from A and one from B, that refer to the same entity in a given domain. The process of finding those correspondences we called instance matching. The result of these mappings we will refer as a RDF interlinking system. This matching process is based on string matching algorithms. Following figure 8.11 presents use interface for the RDF linking datasets.

Figure 8.11: Web Interface for RDF Data sets Linking

User can select a local RDF file to be uploaded to server. On submission of request to upload the file, our developed system will upload the file into a directory in the server. After that when user click the match button, pipe engine creates pipes to fetch, mix, and process RDF files published on the Web. Then user can download the RDF file with owl:sameas relation and following Figure 8.12 depicts example of interlinking where WordNet RDF datasets interlink with DBpedia and Figure 8.13 describes SEPREMO data sets interlink with DBPedia.

```
<rdf:Description rdf:about="http://www.w3.org/2006/03/wn/wn20/instances/Bowling">
    <sameAs xmlns="http://www.w3.org/2002/07/owl#" rdf:resource="http://dbpedia.org/resource/Bowling"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.w3.org/2006/03/wn/wn20/instances/Communication">
    <sameAs xmlns="http://www.w3.org/2002/07/owl#" rdf:resource="http://dbpedia.org/resource/Communication"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.w3.org/2006/03/wn/wn20/instances/Entrance">
    <sameAs xmlns="http://www.w3.org/2002/07/owl#" rdf:resource="http://dbpedia.org/resource/Entrance"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.w3.org/2006/03/wn/wn20/instances/Article">
    <sameAs xmlns="http://www.w3.org/2002/07/owl#" rdf:resource="http://dbpedia.org/resource/Article"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.w3.org/2006/03/wn/wn20/instances/Absolute_space">
    <sameAs xmlns="http://www.w3.org/2002/07/owl#" rdf:resource="http://dbpedia.org/resource/Absolute_space"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.w3.org/2006/03/wn/wn20/instances/Masterstroke">
    <sameAs xmlns="http://www.w3.org/2002/07/owl#" rdf:resource="http://dbpedia.org/resource/Masterstroke"/>
</rdf:Description>
```

Figure 8.12: WordNet RDF dataset interlink with DBpedia

```
<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/Isolator">
    <sameAs xmlns="http://www.w3.org/2002/07/owl#" rdf:resource="http://dbpedia.org/resource/Isolator"/>
</rdf:Description>

<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/Hammer">
    <sameAs xmlns="http://www.w3.org/2002/07/owl#" rdf:resource="http://dbpedia.org/resource/Hammer"/>
</rdf:Description>

<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/Document">
    <sameAs xmlns="http://www.w3.org/2002/07/owl#" rdf:resource="http://dbpedia.org/resource/Document"/>
</rdf:Description>

<rdf:Description rdf:about="http://earthquake.linkeddata.it/resource/Device">
    <sameAs xmlns="http://www.w3.org/2002/07/owl#" rdf:resource="http://dbpedia.org/resource/Device"/>
</rdf:Description>
```

Figure 8.13: SEPREMO RDF datasets interlink with DBpedia.

The vision of the Semantic Web requires a system that can change data and reuse exchanged data with their intended meanings. This is called semantic interoperability. Experiments conducted with different RDF data sets demonstrate that our approach considerably performs state-of-the-art automatic approaches for solving the interlinking and interoperability problem on the Linked Data Cloud.

8.2.6 Intelligent Query System

Apache Lucene is a widely used text-indexing and searching library; SEISMIC Engineering domain leverages the features of the Lucene search engine library. The basic idea is to gather different kinds of information of the source ontology in Lucene

documents that will be stored into an index. Mappings are discovered by using the values of entities in the target ontology as search arguments against the index created from the source ontology. In particular, similarities between documents in the index and queries are computed by exploiting the scoring schema implemented. Figure 8.14 presents intelligent query systems

Figure 8.14: Intelligent Query System

Moreover, users can search the documents; can specify which field they want to search. Search returns multiple related results. If the two entities are the same, to extract relevant entities we need to look in Wordent for the corresponding entity form the target ontology. The Indexing with Lucene includes three main features:

- Converting data to text: in this phase Lucene converts data (e.g., pdf, doc, ppt, and xls documents) into textual form through the use of appropriate parsers.
- Analyzing the text: in this phase, stop words are eliminated and words are stemmed.
- Saving the text into an index. Lucene can create two types of indexes: one maintained in main memory and the other on the hard disk.

However, when dealing with a diverse set of documents, the indexing schema can get complicated. Our current implementation spans three information domains, namely entity, relation and attribute. As we make progress, we intend to include other information sources such as scientific publications and experimental reports. The tool

provides features such as integration with domain knowledge and with free text and search libraries such as Apache Lucene and Solr.

8.3 RELUIS Database

RELUIS targeted at creating an Italian platform for wide sharing of experimental data and knowledge amongst different university, research and industry, which could be maintained and enhanced over time. Typically, Italian earthquake engineering laboratories generates large amounts of data either in the experimental facilities (shake tables, centrifuges, reaction walls), or by outdoor tests. Few laboratories had adopted the approach of a database for storing their test results, with the majority saving data in a fragmented and unstructured way and without any strategy. Therefore, the dissemination of experimental was problematic. To overcome this scenario, the University of Trento takes an initiative to develop a prototype database and interface for the National Italian laboratory, see Figure 8.15, in this respect.

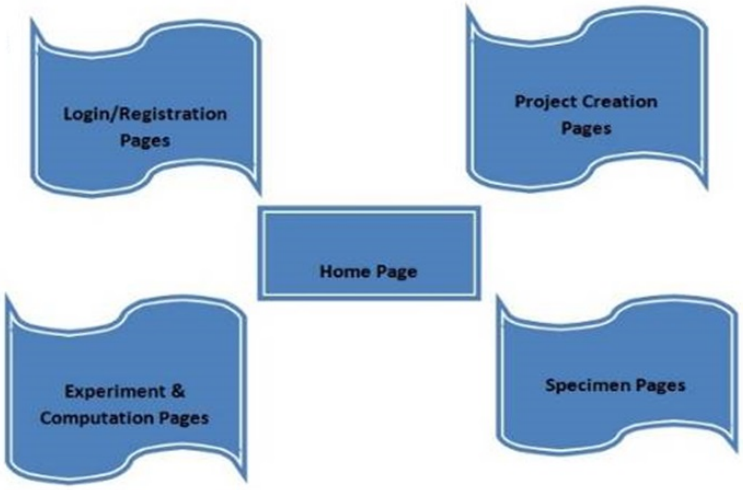


Figure 8.15: The organization of the views of RELUIS application

The interface was designed to enable:

- Login pages

- Database access
- Management of local users

Login pages, collects all pages related to user authentication and registration. They contain pages for user confirmation, mailer, passwords, registration and sessions. Figure 8.16 presents the login page:

Figure 8.16: Login Page

Database access: functionalities to interact with the whole database internal structures in a user-friendly way. Users just need to use a visually appealing interface to create, edit or delete elements in the database without knowing how the database is actually implemented. They can also conduct other tasks such as visualise or search for data. Figure 8.17 depicts a new project page:

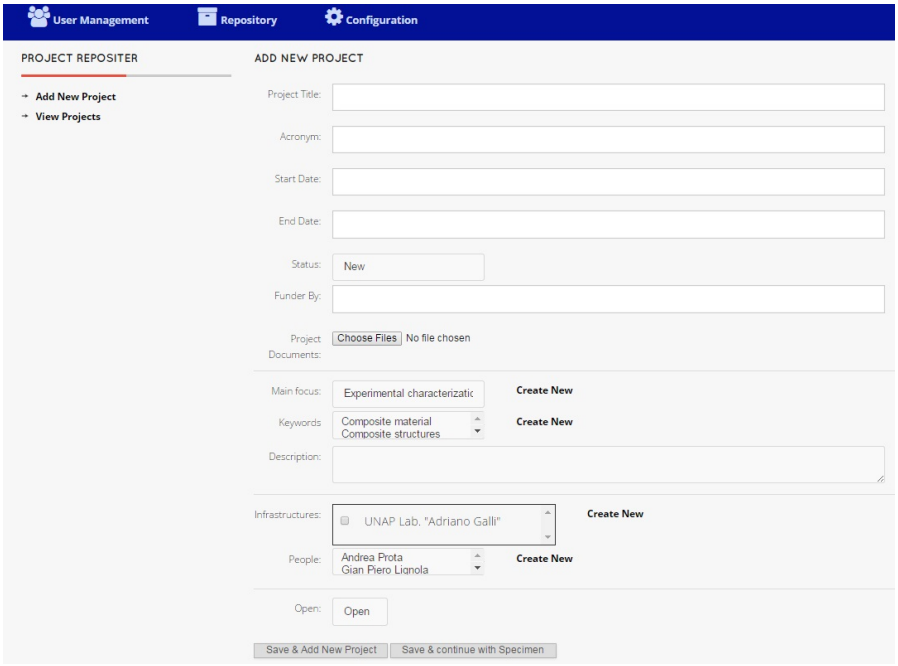


Figure 8.17: Add New Project

RELUIS database can be accessed by an external user at the Data Access Portal (DAP). The interface simulates that of the RELUIS portal with the difference of a left column, which actually presents a breakdown list of available test results from the laboratories participating in RELUIS. Information about the RELUIS database and its format and well as a users manual is available at this level. Navigation around all available data produced by RELUIS laboratories and flagged by them as public, is open without restriction at any level. The information offered may be characterized as general (information about the project and contributors see Figure 18), or detailed (when referring to Specimen, Experiment, Computation or Signals level Figure.8.19-8.22).

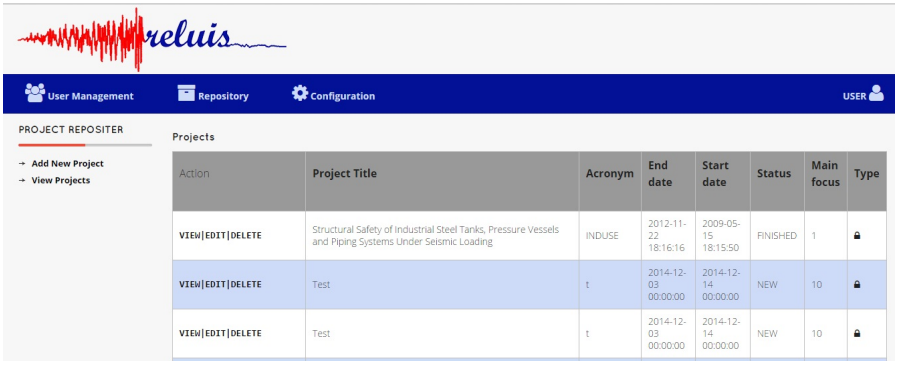


Figure 8.18: List of Projects

Actual data of any type may be freely downloaded for all public project data. If a project is to be accessed only by RELUIS partners, downloading requires user authentication (managed at the Data Access Portal). Nevertheless, regardless of the data type being downloaded from the database, acceptance of the Terms and Conditions displayed is a prerequisite. The statement declares that all intellectual property rights in the data, including, but not limited to, copyright and database rights are vested in their respective right holders.

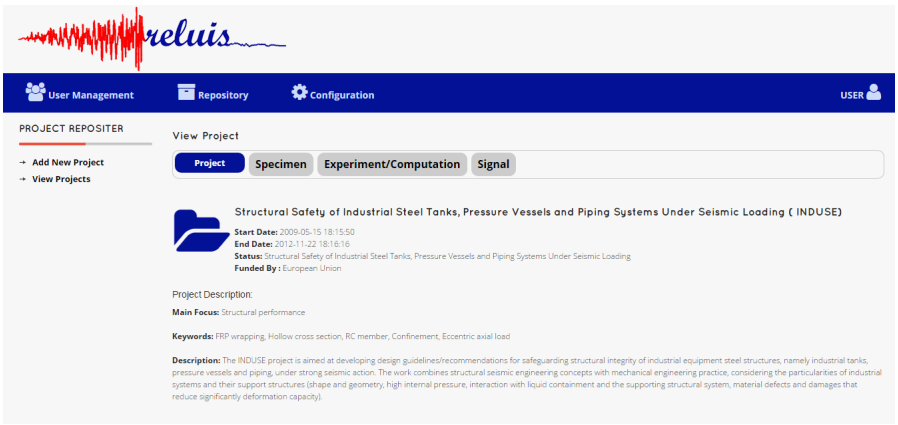


Figure 8.19: Navigation at the Central Site: project level

The DAP is further equipped with a Search functionality which performs a keyword-based search. The keywords forming the basis for the search are presented in categories according to the level they belong to. When more than one filter is selected

they are logically connected by an AND operator, while multi-selection is also allowed for some of the filters.

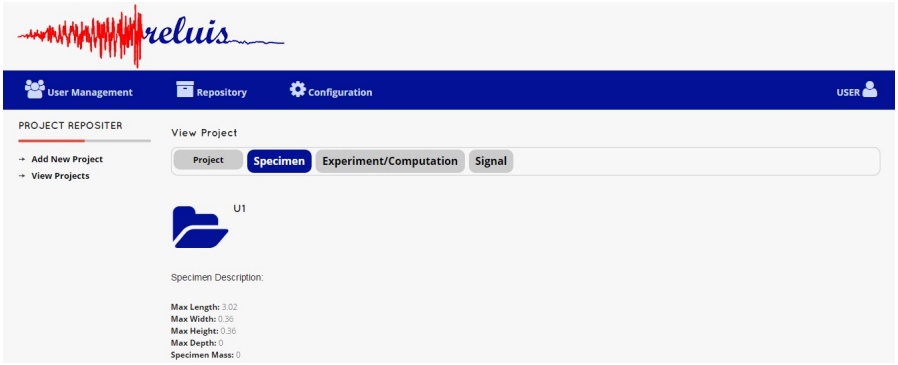


Figure 8.20: Specimen level: expanded view of specimen characteristics

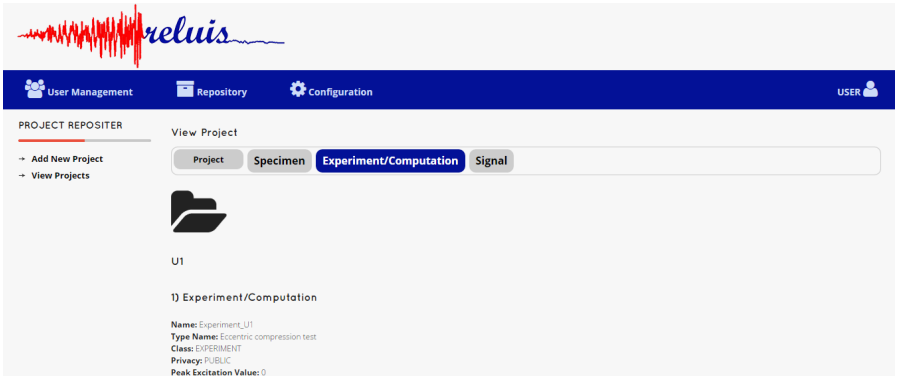


Figure 8.21: Experiment level: tests have been performed

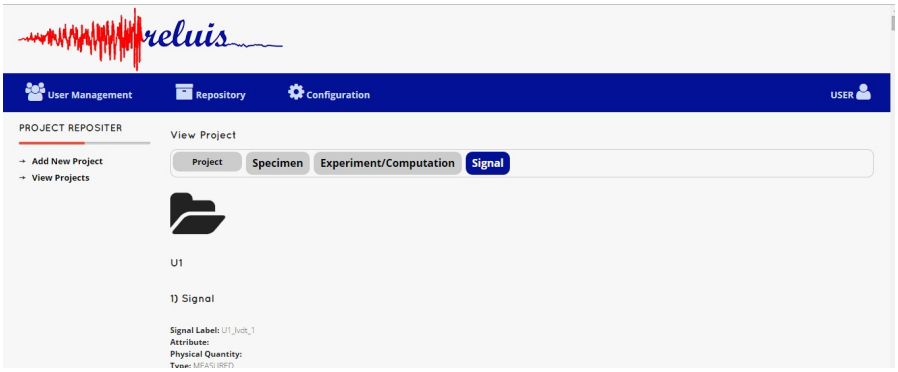


Figure 8.22: Signal level: each signal is delivered together with data regarding its units, the nature of the signal

Management of local users: Web interface allows different local users to access the database. Every user has a role assigned (administrator, user and guest) that enables them to use different functionalities within the interface. For instance, guest users can only visualise data, but they cannot modify any information. Figure 8.23 represents the list of users who are connected in the web interface:

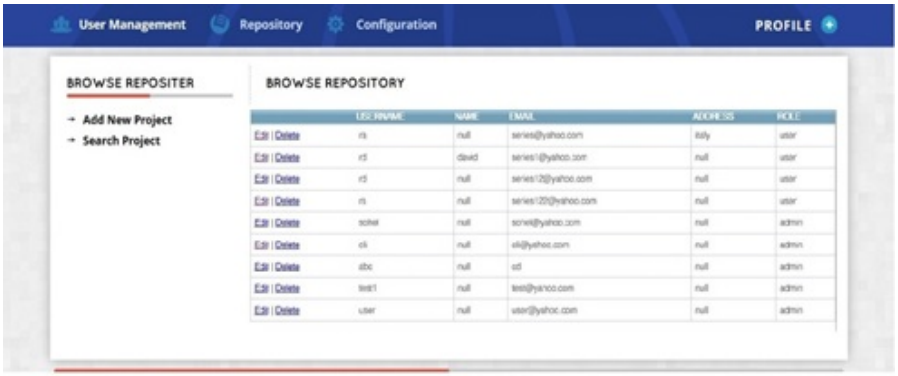


Figure 8.23: List of Users

In the following, Figure 8.24 presents that the user is able to search by user name:

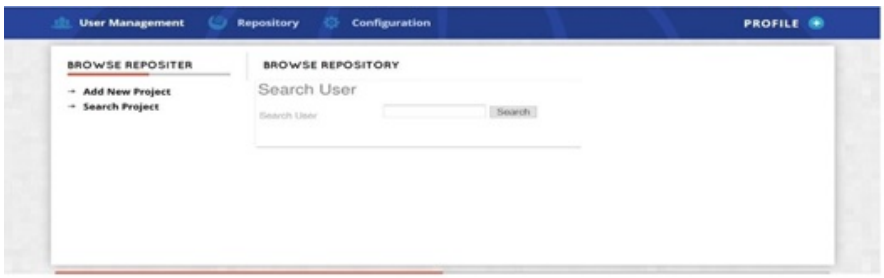


Figure 8.24: Search User Page

According to user role, the user can update profile logo. Along this line, Figure 8.25 represents the interface

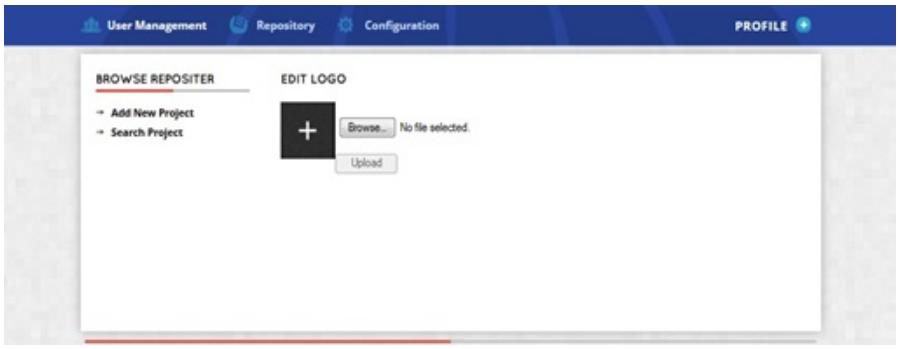


Figure 8.25: Profile Logo Change

The development of the Database represents also a useful support for the development of testing activities. An experiment normally goes through a clearly separated number of phases, highlighting the participation of researchers, computers and laboratory facilities. In most of the stages, human participation is required.

- During the test organization, participants discuss about the experiment objective and the resources that will participate in the experiment. They also have to agree on the data structures that would be exchanged during the test and have to exchange information such as network addresses among other details.

- Once the resources of the experiment are clear, a resources booking phase takes place. This guarantees that the resources will be available when the

experiment is conducted.

- When all participant laboratories are ready, an experiment preparation phase might take place. During this phase, the resources are prepared by the participants and a formal or informal workflow or protocol might be followed.
- The experiment is run in the experiment execution phase. Sometimes, no user interaction is required or desired at this stage. The devices and machines execute the orders given until the test is finished.
- The results are collected by the researchers and a result interpretation is made. This can take an arbitrary long time.
- When the results and conclusions are ready, they can go through the result storage / sharing phase. This allows re-use of work by others, enables machines to operate the data and establishes a working methodology.

8.4 Conclusion

In this Chapter,we provided a detailed description of the development of Earthquake Engineering Projects and Experiments ontology. We followed DERA methodology for building this domain specific ontology. We exploited an ontology integration algorithm that was employed to incorporate our ontology into WordNet. It helped to increase the coverage of the Knowledge Base. On top of the integrated ontology that is kept in an instance of Vrituoso, we experimented the semantic and ontological capabilities of the developed system and interesting results were found. Moreover, we also evaluate how to integrate RDF data sets with LOD cloud that extending the pipes for Earthquake engineering research projects and experiments to build up Mashup. The Mashup enables the combination of existing data sources in the pipes engine, e.g., shakers, devices. Finally, we match every concepts to return sameas relation; we also verified the relationships for accuracy. Intelligent query system search over the documents (i.e. keyword queries matched against bag-of-words document representation) to semantically tagged natural text. By indexing the annotated quotations, users can also search for key words about an entity or a category of entities.

Moreover, the chapter describes the principle and associated elements which con-

stitute RELUIS database. An Exchange Data Format that could host heterogeneous experimental data and provide all the information needed to reproduce a test, has been developed and agreed.

Data stored at local sites is made accessible to external users by means of the Data Access Portal hosted at the University of Trento. In this way a centralized access to database nodes that are distributed over a network and are able to dialog with a central portal in a uniform manner, is provided.

RELUIS database enables a wider sharing of data and knowledge and ultimately, offers an unprecedented service to the earthquake engineering community. RELUIS users will be able to have access to a wide database of experimental data and information, without violating the ownership of the data that will remain with the local laboratory where data have been produced. This platform is to be maintained and enhanced well beyond the end of RELUIS project, to serve as a reference point for the earthquake engineering community worldwide.

CHAPTER 9

CONCLUSIONS

This chapter concludes this thesis by, firstly, providing a brief summary of the work done and then, addressing the limitations of the approach. Finally, it discusses potential directions for further development and presents research conclusions.

9.1 Thesis Summary

In this thesis, the need for ontologies in Seismic Engineering is discussed, and it has been shown that ontology can be a useful tool for knowledge sharing and reuse. We have investigated the nature, construction and practical role of ontologies as mechanisms for knowledge sharing and reuse in the application. Hence, developing ontologies is an important aspect of the Semantic Web. The development of an ontology called EERP that covers a wide range of experiments and projects was described. It showed in detail that the DERA methodology can be employed to gradually develop large domain ontologies in a structured fashion. In this case, we have used and reused generic ontologies, and this approach enhances both the modularity and the reusability of ontologies. We have also introduced the difficulties associated with the reuse of ontology. During the first phase of the work, we collected terms used by seismic engineering researchers for explaining projects and experiments. We exploited an ontology integration algorithm that was employed to incorporate SEPREMO ontology into WordNet. It helped to increase the coverage of the KB. The SEPREMO and its integration framework could enable researchers of Seismic Engineering systems to link engineering information from different sources, such as

projects, the RELUIS database, and experiments. This would help researchers and organizations in different scenarios:

- searching for Seismic engineering information (e.g., it could facilitate automated mapping of researcher-entered queries to technical terms);
- translating and interpreting projects information or test results;
- describing their experimental history and their complaints;
- involvement of people and organizations in the particular project;
- visualization of the full ontology through web;
- Sparql Endpoint for the expertise.

Moreover, each kind of concept has been described in detail and implemented within the developed ontology. We also collected data that consist not only the terms to be included in the Vocabulary but also in synonyms for them, descriptions for each project and experiment. At this stage we studied which resources to use in our tasks and how to use each of them, how to discover instances from those resources, how to link their content and understand the content of the resources and interpret the results. However, terms are generally formed from noun phrases; in some cases verbs may also be considered, but we ignored this in the developed ontology. Term recognition has been performed on the basis of various criteria; using Wikipedia, WordNet and possible sources of earthquake engineering documents as a knowledge source for extraction of terms. This vocabulary was organized as a hierarchical structure that established relationships among its terms and concepts. This can be done largely automatically, semi-automatically or manually with the help of domain experts and published in semantic web language, namely RDF and OWL. The vocabulary by focusing on its content and semantics, independent of any application, produces a representation that is suitable for a wide variety of applications. The resulting ontology covers two main areas of the domain knowledge: projects and experiments. On top of the integrated ontology that is kept in an instance of Vrituoso, we experimented the semantic and ontological capabilities of the developed system and interesting results were found. Finally, by defining some application scenarios, we tried to show,

what a seismic engineering application can gain from the use of ontology-based technologies. Starting from a classification of programming languages and approaches that aim to formalize them, we discussed approaches that allow for dealing with web based systems. In addition, computability and complexity of system structures were also discussed.

We have discussed the current situation in the ontology matching domain with particular emphasis on the significant role of matching approaches in the realization of the Semantic Web vision. We have analyzed the main open issues and considered what kind of methodological and tool support is needed to cover the gaps. Therefore, another goal of this research was to solve the problem of ontology alignment, thus enabling semantic interoperability between different applications. In the field of ontology matching, one of the main issues is the need for algorithms and tools, capable of adapting to different domains and also to different interpretations of the notions of alignment and similarity. Our system implements a normalization method that is based on syntactic matching in order to provide an automatic alignment framework for the purpose of improving semantic interoperability in heterogeneous systems. Such ontology alignment means linking entities of source ontology with those of target ontology based on different features of these ontologies and using different strategies. For the end-user, this work provided an easy-to-use tool for ontology alignment.

This thesis also inspected the issue of managing large number of files and proposed a system for facilitating the file retrieval. Matching degrees were defined in order to match the relevant keywords that exist in the file while searching for a required file. We used Apache Lucene that provides search over documents. A document is essentially a collection of fields, where a field supplies a field name and value. Moreover, our system manages a dynamic document index, which supports adding documents to the index and retrieving documents from the index using search API. The index structure provides the reverse mapping from terms, consisting of field names and tokens, back to documents. To search this index, we construct a term composed of the field title and the tokens resulting from applying the key words that listing to the text we are looking for.

At the end, this work provides some highlights of what we did in this period of research and implements some of the possible solutions among the available ones. We have also shown how ontologists could develop domain ontologies merging different methodologies and software engineering techniques, taking advantages of them. Particularly, this approach has been used to define a Domain Ontology for a Seismic Engineering, which could be extended and used by different research applications.

9.2 Limitations

A limitation of this thesis is that the ontology generation may become a time consuming process. Manual creation of software system ontology based on design principles may become too time consuming and may even become a burden of the project. There is a right balance between manually creating software system ontology and reusing existing software system ontology. However, top-down approach may still become too complicated and time consuming.

9.3 Future Work

No domain is capable of performing perfect modelling since specifications vary from one application to another, and the future extensions might have new requirements. Apart from adding new concepts, sharing, reusing, maintaining and evolving, which are important issues for the future, each module has its own characteristics that can be improved. For example, a ranking based classification would increment the information level about projects and experiments. The application could include a personalized configuration to specify the results that the user can edit and assign the rank of each entity. An improvement to the actions module could be the integration of a process ontology to define complex actions. Such restructuring should be carefully thought, since the benefits may be outweighed by difficulties. Moreover, the storing mode of data and the ordering of retrieval results need to be improved. It will influence the system capability seriously when the knowledge base and ontology model enlarge to a certain degree.

The ontology matching approaches developed in this thesis did not contemplate all identified issues, so these are obvious targets for future work. Besides, our system only supports syntactic matching and it does not support selecting external resources, which are crucial to correctly identify some matches for which there is no support in the ontology information alone. We would like to reduce human intervention in our transformation system to the minimal unavoidable cases; this can be achieved by identifying the cases that can be converted from manual to automatic without compromising the quality of the system's result. It may be possible to improve the performance of the algorithm itself and of its implementation in order to deal with very large files. We are also interested in applying ontology alignment techniques to issues related to the disaster concerns of Big Data. Currently, many linked datasets are anonymized before being made available on the Semantic Web.

Various models of information retrieval have been developed over the past years. Now search results cannot be ranked appropriately because the documents are identified as relevant by matching to the query. We will improve our work and facilitate users get documents rank of relevancy. This approach can improve the retrieval performance by its ranking schema which can improve extracting performance. In addition, our ongoing research involves improvement of querying capabilities and using Natural Language Processing (NLP) techniques for ontology update. Moreover, Linked Data has been recently suggested as one of the best alternatives for creating shared information spaces. In the context of Linked Data, the RDF language is used to describe resources in the form of triples. One extension of the work of this thesis is the generation of RDF data following the Linked Data principles.

9.4 Conclusions

This thesis has developed an interdisciplinary ontology in a semi-automated fashion. Seismic Engineering ontology can play a progressively important role in Civil engineering as well as in Mechanical and Environmental Engineering in general. As Seismic engineering become increasingly data driven, the need to add a semantic layer to these large collections of data becomes more pressing. Application of Seismic Engineering otology is becoming more prevalent in diverse areas such as, search and query heterogeneous projects and experiments data, data exchange among ap-

plications, information integration, NLP and reasoning with data. This is a highly demanding task, especially in an active and complex domain like the seismic engineering field. The future ontology development will necessarily incorporate the automation of some of its processes, mainly those that are tedious and time-consuming.

BIBLIOGRAPHY

A. Bosi, I. Kotinas, I. Lamata Martnez, S. Bousias, J.L. Chazelas, M. Dietz, Hasan M. R., S.P.G. Madabhusi, A. Prota, T. Blakeborough, P. Pegon, 2013. The SERIES Virtual Database: Exchange Data Format and local/central databases. SERIES Concluding Workshop - Joint with US-NEES, Earthquake Engineering Research Infrastructures JRC-Ispra, May 28-30, 2013

Denicola, a., Missikoff, M. & Navigli, R. , 2009. A software engineering approach to ontology building. Information Systems. Information Systems, 34, 258-275

Sure, Y., Staab, S., Studer, R. & Gmbh, O. , 2003. Handbook on Ontologies, International Handbooks on Information Systems, chap. On-to-knowledge methodology (OTKM) Springer-Verlag,117-132.

Giunchiglia, Fausto, Pavel Shvaiko, and Mikalai Yatskevich, 2003. S-Match: an algorithm and an implementation of semantic matching Springer Berlin Heidelberg, 2004

Giunchiglia, Fausto, and Biswanath Dutta, 2011. DERA: A faceted knowledge organization framework 2011

Farazi, F., Maltese, V. , Giunchiglia, F. , Ivanyukovich, A, 2011. A faceted ontology for a semantic geo-catalogue In The Semanic Web: Research and Applications (pp. 169-182). Springer Berlin Heidelberg.

Le-Phuoc, D., Polleres, A., Tummarello, G., & Morbidoni, C. , 2008. DERI pipes: visual tool for wiring web data sources

Warrington, John , 1956. Aristotle's metaphysics London: JM Dent; New York: EP Dutton, 1956

Hepp, Martin, 2008. GoodRelations: An Ontology for Describing Products and Services Offers on the Web. 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008), Acitrezza, Italy, 2008,vol. 5268, pp. 332-347.

Davies, John, and Richard Weeks, 2004 QuizRDF: Search technology for the semantic web. System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on. IEEE, 2004.

Hasan, M.R., Farazi, F., Bursi, O.S., Reza, M.S., 2013 In A Semantic Technology-Based Information Management System for Earthquake Engineering Projects and Experiments. 5th International Conference on Advances in Experimental Structural Engineering, Taipei, Taiwan: National Center for Research on Earthquake Engineering, 2013

Fensel, Dieter. , 2001. Ontologies Springer Berlin Heidelberg,2001

Gruber, Thomas R.,1993. A translation approach to portable ontology specifications Knowledge acquisition 5.2 (1993):199-220.

Giaretta, Pierdaniele, and N. Guarino.,1995. Ontologies and knowledge bases towards a terminological clarification.Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing 1995 25-32.

Falbo, R. A., Guizzardi, G., Duarte, K. C, 2002 An Ontological Approach to Domain Engineering. Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE), Ischia, Italy, 2002.

Gmez-Prez, A., Fernndez-Lpez, M., Corcho, O. 2004. Ontological Engineering. Springer Verlag,Berlin, 2004.

Fernandez-Lopez, Mariano, Asuncion Gomez-Perez, and Natalia Juristo, 1997 Methontology: from ontological art towards ontological engineering.

A. Gomez-Perez, O. 1994. Some Ideas and Examples to Evaluate Ontologies Technical Report KSL-94-65, Knowledge Systems Laboratory , Stanford, 1994.

AY. Sure, S. Staab, and R. Studer, 2003. On-To-Knowledge Methodology (OTKM). Handbook on Ontologies. Springer, 2003.

Baader, F., Calvanese, D., McGuinness, D, Nardi, D., Patel-Schneider, P. F., 2002 The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2002)

Ranganathan, S. R. , 1967 Prolegomena to library classification. Asia Publishing House,1967.

Klyne G., Carroll J., 2004 Pesource Description Framework (RDF): Concepts and Abstract Syntax W3C Recommendation, 10 February 2004.

Steer D, Miller L, Brickley D, 2002 RDFAuthor: Enabling everyone to author rdf WWW02 Developers Day, 2002

Giunchiglia, F., Farazi, F., Tanca, L. , R. de Virgilio, 2010 The Semantic Web Languages. Semantic Web Information Management Springer Verlag, Berlin, 2010.

Euzenat, Jrme, and Pavel Shvaiko, 2007 Ontology matching Vol. 18. Heidelberg: Springer, 2007.

Castano, S., V. De Antonellis, and S. di Vimercati, 2000. Global Viewing of Heterogeneous Data Sources In Transactions on Knowledge and Data Engineering, 13(2): 277-297. IEEE.

Madhavan, J., P. Bernstein, and E. Rahm, 2001. Generic Schema Matching with Cupid In 27th International Conference in Very Large Databases, 49-58. Rome (IT).

Ehrig, M., and Y. Sure. 2005. FOAM - Framework for Ontology Alignment and Mapping; Results of the Ontology Alignment Initiative In Proc. K-CAP Workshop on Integrating Ontologies, 156: 72-76. Banff (CA).

Euzenat, J. 2004. An API for Ontology Alignment In Semantic Web - ISWC 2004, LNCS 3298: 698-712. Springer

Straccia, U., and R. Troncy. 2005. oMAP: Combining Classifiers for Aligning Automatically OWL Ontologies In Semantic Web - ISWC 2004, LNCS 3298: 698-712. Springer

P. Hitzler, R. Sebastian, and M. Krotzsch, 2009. Foundations of Semantic Web Technologies Chapman & Hall/CRC, London, 2009.

Athanasis N, Christophides V, Kotzinos D, 2004. Generating On the FlyQueries for the Semantic Web ISWC2004.

Russell A, Smart R, Braines D, Shadbolt R, 2008. NITELIGHT: A Graphical Tool for Semantic Query Construction SWUI Workshop. 2008.

T. Berners-Lee, 1999. Weaving the Web Orion Business Books, 1999

T. Berners-Lee, J. A. Hendler, and O. Lassila, 2001. The Semantic Web In Scientific American Journal,34-43,2001

Heflin, J. and Hendler, J. , 2001. Dynamic ontologies on the web In Proceedings of AAAI 2000. AAAI Press

Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H., and Sure, Y., 2005 A framework for handling inconsistency in changing ontologies In Proceedings of ISWC05, volume 3792 of LNCS, pages 353367. Springer

Bobillo, F. and Straccia, U. , 2008 An expressive fuzzy description logic reasoner In In Proceedings of FUZZ-08.

Peng, Y., Ding, Z., and Pan, R. , 2005 A probabilistic framework for uncertainty in semantic web. In Proceedings of Nineteenth International Joint Conference on Artificial Intelligence (IJCAI05).

B. Kapoor and S. Sharma, 2010 A Comparative Study Ontology Building Tools for Semantic Web Applications. International Journal of Web & Semantic Technology (IJWesT), Vol. 1, No. 3, pp. 1-13, July, 2010

M.R. Khondoker and P. Mueller, 2010 A Comparative Study Ontology Building Tools for Semantic Web Applications. International Journal of Web & Semantic Technology (IJWesT), Vol. 1, No. 3, pp. 1-13, July, 2010

H. Knublauch, R. W. Fergerson, N. F. Noy and M. A. Musen, 2004 The Protege OWL Plugin: An Open Development Environment for Semantic Web Applications. In

Proceedings of the Third International Semantic Web Conference, Lecture Notes in Computer Science, Hiroshima, Japan, November 7-11, pp. 229-243, 2004

C. Calero, F. Ruiz and M. Piattini, 2006 Ontologies for Software Engineering and Software Technology. Calero.Ruiz.Piattini (Eds.), Springer-Verlag Berlin Heidelberg, 2006

K. Wilkinson, C. Sayers, H. Kuno and D. Reynolds, 2003 Efficient RDF Storage and Retrieval in Jena2. In proceedings of the First International Workshop on Semantic Web and Databases (SWDB), Berlin, Germany, pp. 131-150, 2003.

M. Watson, 2008 Practical Artificial Intelligence Programming with Java. Third Edition, 11 November, pp. 57-72, 2008.

R. Ihaka, R. Gentleman, 1996 R: A language for data analysis and graphics, Journal of Computational and Graphical Statistics. 5 (3) (1996) 299314.

R. Gentleman, 2008 R Programming for Bioinformatics. Chapman & Hall/CRC, 2008, iSBN 978-1-420-06367-7.

Thusoo, Ashish, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu, and Raghotham Murthy, 2010 Hive-a petabyte scale data warehouse using hadoop. Data Engineering (ICDE), 2010 IEEE 26th International Conference on. IEEE, 2010.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. , 2010 Spark: cluster computing with working sets. In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing.

Giunchiglia, F., Dutta, B. , 2011 DERA: A Faceted Knowledge Organization Framework. March 2011

Dutta, Biswanath, Fausto Giunchiglia, and Vincenzo Maltese , 2011 A facet-based methodology for geo-spatial modeling GeoSpatial Semantics. Springer Berlin Heidelberg, 2011. 133-150

M. Uschold, M. Healy, K. Williamson, P. Clark, and S. Woods , 1998 Ontology reuse and application FOIS98, Treno, Italy, 1998

Farazi, F., Maltese, V. , Giunchiglia, F. , Ivanyukovich, A, 2011 A faceted ontology for a semantic geo-catalogue Extended Semantic Web Conference (ESWC), 2011

F. Giunchiglia, P. Shvaiko, and M. Yatskevich, 2005 Semantic Schema Matching In Proceedings of OTM Conferences (1), 2005, pp.347-365

AnHai Doan, Pedro Domingos, Alon Halevy, 2003 Learning to Match the Schemas of Data Sources: A Multistrategy Approach Machine Learning, 50 (3): 279- 301, March 2003

G. A. Miller, F. Hristea, 2006 WordNet Nouns: classes and instances Computational Linguistics, 32(1):1.3 (2006) Cruse, D.A. 1986. Lexical Semantics. New York: Cambridge University Press

Flouris, G., Manakanatas, D., Kondylakis, H., Plex-ousakis, D. & Antoniou, G. , 2008 Ontology change: classification and survey The Knowledge Engineering Review

Stojanovic, L., Maedche, A., Motik, B. & Stojanovic, N. , 2002 User-driven Ontology Evolution Management. Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW-02), Lecture Notes in Computer Science (LNCS), Volume 2473, Springer- Verlag, pp, 285-300.

Stojanovic, L., Maedche, A., Motik, B. & Stojanovic, N. , 2003 Ontology Evolution as Reconfiguration-Design Problem Solving. In Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP-03), pages 162-171, 2003.

M. Klein, D. Fensel, 2001 Ontology Versioning on the Semantic Web. In Proceedings of the International Semantic Web Working Symposium (SWWS), pages 75-91, 2001.

P. Haase, Y. Sure. , 2004 D3.1.1.b State of the Art on Ontology Evolution. 2004

P. Plessers, O. De Troyer , 2005 Ontology Change Detection Using a Version Log. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), pages 578-592, 2005

P. Breche, M. Woerner , 1995 How to remove a class in an ODBS. In Proceedings of the 2nd International Conference on Applications of Databases (ADBS95), San Jose, California, pp. 235-246, 1995.

Giunchiglia, Fausto, Vincenzo, Maltese, Feroz, Farazi, and Biswanath, Dutta, 2010. GeoWordNet: a resource for geo-spatial applications. In The Semantic Web: Research and Applications, pp. 121-136. Springer Berlin Heidelberg, 2010.

D. Crystal, 1980. A first dictionary of linguistics and phonetics. Westview Press, Boulder, CO, 1980.

Nirenburg, Sergei, and Victor Raskin, 2004. Ontological semantics. MIT Press, 2004.

M.R. Hasan, F. Farazi, O.S. Bursi and M.S. Reza, 2014. Towards Publishing Earthquake Engineering Experimental Data As Part Of Linked Open Data Cloud. In The 8th INSPIRE Conference, Aalborg, Denmark, 2014

Brickley, D., Guha, R., and Layman, A. , 1998. Resource description framework (RDF) schema specification. Working draft, W3C. http://www.w3c.org/TR/WD-rdf-schema.

Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks, 2004. OWL Web Ontology Language Semantics and Abstract Syntax section 5. rdf-compatible model-theoretic semantics. W3C Recommendation, http://www.w3.org/TR/owl-semantics/rdfs.html, February 2004.

PG. Antoniou and F. Van Harmelen, 2004. Web Ontology Language: OWL. In Handbook on Ontologies. pages 91110. Springer, 2009.

Grau, Bernardo Cuenca, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler, 2008. OWL 2: The next step for OWL. Web Semantics: Science, Services and Agents on the World Wide Web 6, no. 4 (2008): 309-322.

B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz, 2012. OWL 2 Web Ontlogy Language profiles (second edition). W3C Recommendation, December 2012.

Phuoc, D.L., Polleres, A., Morbidoni, C., Hauswirth, M., Tummarello., G., 2009. Rapid Semantic Web Mashup Development Through Semantic Web Pipes. Proceedings of the 18th World Wide Web Conference (WWW2009). ACM Press, Madrid, Spain (2009), pp. 581-590.

Volz, J., Bizer, C., Gaedke, M., & Kobilarov, G., G., 2009. Silka link discovery framework for the web of data. In Proceedings of the 2nd Linked Data on the Web Workshop (LDOW2009), Madrid,Spain, 2009

Dunne, M., E. I. Moses, P. Amendt, T. Anklam, A. Bayramian, E. Bliss, B. Debs , 2011. TIMELY DELIVERY OF LASER INERTIAL FUSION ENERGY(LIFE). Fusion Science and Technology 60, no. 1 (2011): 19-27.

Morbidoni, Christian, Axel Polleres, Giovanni Tummarello, and Danh Le Phuoc , 2007. Semantic web pipes. IDA Business Park, Lower Dangan, Galway, Ireland (2007).

Soren Auer and Jens Lehmann , 2010. Creating knowledge out of interlinked data. Semantic Web Journal, 1(1,2):97-104, 2010.

Soren Auer, 2011. Creating knowledge out of interlinked data: making the web a data washing machine. In Proceedings of the International Conference on Web Intelligence, Mining and Semantics, pages 4:1-4:8, New York, USA, 2011. ACM.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z., 2007. Dbpedia: A nucleus for a web of open data. Sixth International Semantic Web Conference, Busan, Korea (2007) 1115

Tim Berners-Lee, 2006. Linked Data - Design Issues.

Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, Renaud Delbru, and Stefan Decker. Sig.ma, 2010. Live views on the web of data. Journal of Web Semantics, 8(4):355-364, 2010

Tim Finin, Yun Peng, R. Scott, Cost Joel, Sachs Anupam Joshi, Pavan Reddivari, Rong Pan, Vishal Doshi, and Li Ding. , 2004. Swoogle: A search and metadata engine for the semantic web. In Proceedings of the 13th ACM Conference on Information and Knowledge Management, pages 652-659. ACM Press, 2004.

Georgi Kobilarov, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee , 2009. Media meets semantic web-how the bbc uses dbpedia and linked data to make connections. In Proceedings of the 6th European Semantic Web Conference, pages 723-737. Springer Berlin Heidelberg, 2009.

Christian Bizer, Tom Heath, and Tim Berners-Lee, 2009. Linked data - the story so far. International Journal on Semantic Web and Information Systems, 5(3):1-22, International Journal on Semantic Web and Information Systems, 5(3):1-22, 2009.

Shvaiko Pavel and Jeroome Euzenat, 2013. Ontology matching: State of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering, 25(1):158-176, 2013.

Tom Heath and Christian Bizer, 2011. Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool, 2011.

Handschuh, S., Staab, S., 2003. Annotation for the Semantic Web. IOS Press, (2003).

Smiley, D., Pugh, E., 2011. Apache Solr 3 Enterprise Search Server. p. 418 (2011).

McCandless, M., Hatcher, E., Gospodnetic, O., 2010. Lucene in Action, Second Edition. Covers Apache Lucene 3.0, p. 475 (2010).

Seeley, Yonik, 2006. Apache Solr. ApacheCon Europe 2006

Reza, Md Shahin, et al., 2013. Pseudo-Dynamic Heterogeneous Testing With Dynamic Substructuring of a Piping System Under Earthquake Loading. ASME 2013 Pressure Vessels and Piping Conference. American Society of Mechanical Engineers, 2013.

M.R. Hasan, F. Farazi, O.S. Bursi, M.S. Reza , 2013. A faceted lightweight ontology for earthquake engineering research projects and experiments. In EU-SERIES Concluding Workshop, 2013.

M.R. Hasan, F. Farazi, O.S. Bursi, M.S. Reza , 2013. A Semantic Technology-Based Information Management System for Earthquake Engineering Projects and Experiments. In 5th International Conference on Advances in Experimental Structural Engineering, Taipei, Taiwan: National Center for Research on Earthquake Engineering, 2013.

APPENDIX A

SEISMIC ENGINEERING RESEARCH PROJECTS AND EXPERIMENTS MANAGEMENT ONTOLOGY FACET

1. Earthquake Engineering Project Facet:

- Project
    - Earthquake Engineering Project
    - Civil Engineering project
    - Mechanical engineering project
    - Environmental engineering project

2. Project person Facet:

- Coordinator
    - Principal Investigator (IS-a head of)
    - Local co-investigator
- Partner (Beneficiary/ Research Unit)

3. Experimental Computation Facility Facet:

- Experimental computation facility
    - Computer system
        - Experimental computational system, Data acquisition system
        - Software-system, Software
            - Database

4. Device Facet:

- Device
    - Shaker
    - Hammer (Instrument)
    - Controller
    - Potentiometer
    - Actuator()
        - Cylinder(part-of)
        - Piston(part-of)
        - Electric actuator
            - Motor(part-of)
    - Active Structural device

- Passive Structural device
- Damper
    - Hydraulic damper(is also is-a passive structural device)
    - Electrical damper(is also is-a passive structural device)
    - Magneto Rheological (MR) damper (semi active damper)
    - Friction damper(is also is-a passive structural device)
    - Tuned mass damper(is also is-a passive structural device)
    - Elastomeric Damper
    - Isolator, Vibration absorber
        - High damping Isolator
        - Seismic base Isolator
    - Viscoelastic damper
    - Metallic damper

- Passive Device
- Isolation Device
    - Slider
    - Elastomeric bearing
    - Lead-rubber bearing
    - Friction pendulum bearing
- Sensor
    - Accelerometer
    - Conductivity Sensor
    - Depth Gage
    - Displacement Sensor
    - Inclinometer
    - Load Cell
    - Position Sensor
    - Pressure Sensor
    - Profile Sensor
    - Temperature Sensor
    - Wave Gage
    - ADV
    - Linear Displacement Transducer
    - Micro ADV
    - Pore pressure transducer
    - Position transducer
    - Potentiometer
    - Pressure sensor
    - Slave wave gauge

- Sonic profile transducer
- Strain gauge
- Teledyne pressure transducer
- Velocimeter

5. Specification Facet:

- Document
  - Nominal property document
  - Device document
  - Structural component document
  - Original load signal document
  - Specimen document
  - Material document
  - Project document
  - Experiment computation document
  - Meshmodel document

6. Experiment Facet:

- Experiment
  - Static test
    - Cyclic test
    - Monotonic test
  - Dynamic test
    - PSD(Pseudo dynamic) test with sub structuring
    - Shaking table test
    - Shaker Based test
    - Hammer Based test
  - Hybrid test
    - Numerical Sub-structure
    - Physical Sub-structure
    - Coupling
    - Decoupling
    - Numerical Simulation
    - Degrees of Freedom reduction

8. Computer File facet:

- Computer File
    - Video
    - Image
        - Meshmodel image
        - Sensor configuration image
        - Experiment computation image

9. Organization Facet:

- Organization
    - Institution
        - Research Institute
        - Company
        - Academic Institute

    - Infrastructure
        - Laboratory
            - Research Laboratory
            - Electrical equipment(part-of)
                - Air conditioning unit
                - Wind turbine
            - Mechanical equipment(part-of)
                - Piping

10. Finite Element Model Facet:

- Framework
    - Mesh model
        - Numerical model
            - Mathematical model
            - Element model (relate)

11. Specimen Facet:

- Specimen
    - Structural Component
            - Bridge
            - Brackets
            - Masonary
            - Studs
            - Connection, connector, joint
                - Expansion Joint

- Seismic Joint
- Element(substance Meronym)
  - Linear Element
    - Beam
      - Concrete Beam
      - Simply support Beam
      - Steel Beam
      - Cantilever
        - Balanced Cantilever
        - Shear Cantilever
    - Column
    - Grider
    - Tendon
    - Cable
    - Trusses
    - Braces
    - Piers
  - Plane Element,2D
    - Slab
    - Deck
    - Wall
    - Shell
    - Membrane
    - Arch
    - Cladding
  - Spatial Element
    - Vault
  - Geotechnical Structure
    - Piles
    - Pile group
    - Retaining wall
    - Foundation
    - Earth structure dam
    - Reinforce soil
    - Tunnel
  - Geological Formation
    - Rock Formation
- Experimental Equipment
  - Centrifuge
  - Large scale

- Shaker
    - Linear Shaker
    - Triaxial Mobile Shaker
    - Uniaxial Shaker
    - Multi-Axial Subassemblage Testing

- Strong Floor
- Strong Wall
- Tsunami wave Basin
- Vibroseis Truck