Ontology Ranking: Finding the Right Ontologies on the Web

Anila Sahar Butt

A thesis submitted for the degree of
Doctor of Philosophy in Computer Science
The Australian National University

August 2017

© Anila Sahar Butt 2017

Except where otherwise indicated, this thesis is my own original work.

Anila Sahar Butt
27 August 2017

to my parents

Acknowledgments

I would like to take this opportunity to express my sincere thanks to those who helped me in one way or another in making my stay pleasant at Australian National University (ANU), conducting this research and the writing of this dissertation.

I would like first to thank my supervisor, Dr. Armin Haller, for every favor he showed to me during these years. He was the first person who received me in Canberra at the end of a long tiring journey from Pakistan. Since then not only approachable, but he always responded to any queries that I had, refined my research ideas, listened to my problems, and supported me every step of the way. He has been a great source of inspiration for me during these years of research. Without his encouragement and constant support, this thesis would not have been possible.

I would like to extend my gratitude to my panel members, especially to Dr. Lexing Xie. She provided me every possible opportunity to tailor and share research ideas within the group and to the larger audiences. I would further like to thank everyone at Computational Media Lab for supporting me and organising all adventurous outdoor activities.

My heartfelt thanks go to The ANU and Commonwealth Scientific and Industrial Research Organisation (CSIRO) for awarding me HDR Merit Scholarship and stipend throughout my Ph.D. candidature. I am thankful to the office and academic staff, especially to Janette Rawlinson, for not letting me worry about anything other than my research. A very special thank to all my friends especially Banda, Narjess, Qiufen, Rani, Richa, Sara, Zakir, and Zichuan, who always stood by my side and supported me through all the weathers. My sincere thanks to all the Australians for being acceptable to different cultures and communities. Huge love for Australia.

I would also like to thank my family who has endured me through this period of research that has been extremely demanding and challenging. Thanks to my mother for always being so motherly in everything she does. Especial thanks to my father for showing his confidence in me and letting me pursue my dreams. To my parents, I dedicate this thesis. I would also like to thank my grandmother, brothers and sister for their love and prayers during all these years. I am immensely grateful to my husband for always being appreciative of my efforts, for his unending support and for encouraging me during the hardest of times.

Last, but not the least, my unlimited prostration to Allah, the all mighty, for keeping me steadfast and blessing me the happiness of doing Ph.D.

Abstract

Ontology search, which is the process of finding ontologies or ontological terms for users' defined queries from an ontology collection, is an important task to facilitate ontology reuse of ontology engineering. Ontology reuse is desired to avoid the tedious process of building an ontology from scratch and to limit the design of several competing ontologies that represent similar knowledge. Since many organisations in both the private and public sectors are publishing their data in RDF, they increasingly require to find or design ontologies for data annotation and/or integration. In general, there exist multiple ontologies representing a domain, therefore, finding the best matching ontologies or their terms is required to facilitate manual or dynamic ontology selection for both ontology design and data annotation.

The ranking is a crucial component in the ontology retrieval process which aims at listing the '*relevant*' ontologies or their terms as high as possible in the search results to reduce the human intervention. Most existing ontology ranking techniques inherit one or more information retrieval ranking parameter(s). They linearly combine the values of these parameters for each ontology to compute the relevance score against a user query and rank the results in descending order of the relevance score. A significant aspect of achieving an effective ontology ranking model is to develop novel metrics and dynamic techniques that can optimise the relevance score of the most relevant ontology for a user query.

In this thesis, we present extensive research in ontology retrieval and ranking, where several research gaps in the existing literature are identified and addressed. First, we begin the thesis with a review of the literature and propose a taxonomy of Semantic Web data (i.e., ontologies and linked data) retrieval approaches. That allows us to identify potential research directions in the field. In the remainder of the thesis, we address several of the identified shortcomings in the ontology retrieval domain. We develop a framework for the empirical and comparative evaluation of different ontology ranking solutions, which has not been studied in the literature so far. Second, we propose an effective relationship-based concept retrieval framework and a concept ranking model through the use of learning to rank approach which addresses the limitation of the existing linear ranking models. Third, we propose *RecOn*, a framework that helps users in finding the best matching ontologies to a multi-keyword query. There the relevance score of an ontology to the query is computed by formulating and solving the ontology recommendation problem as a linear and an optimisation problem. Finally, the thesis also reports on an extensive comparative evaluation of our proposed solutions with several other state-of-the-art techniques using real-world ontologies. This thesis will be useful for researchers and practitioners interested in ontology search, for methods and performance benchmark on ranking approaches to ontology search.

List of Publications

- **RecOn: Ontology Recommendation for Structureless Queries:** *Anila Sahar Butt*, Armin Haller and Lexing Xie. In Applied Ontology Journal, vol. 11, no. 4, Pages 301-324, Feb 2017. *Corresponds to Chapter 8 of this thesis.*
- **DWRank: Learning concept ranking for ontology search:** *Anila Sahar Butt*, Armin Haller and Lexing Xie. In Semantic Web Journal, vol. 7, no. 4, Pages 447-461, May 2016. *Corresponds to Chapters 6 and 7 of this thesis.*
- **A Taxonomy of Semantic Web Data Retrieval Techniques:** *Anila Sahar Butt*, Armin Haller and Lexing Xie. In Proceedings of the 8th International Conference on Knowledge Capture (KCAP'15), Pages 7-10 October 2015, Palisades NY, USA. *Corresponds to Chapter 4 of this thesis.*
- **Ontology Search: Finding the Right Ontologies on the Web:** *Anila Sahar Butt*. In Proceedings of the 24th International Conference on World Wide Web Companion (WWW'15), Pages 487-491, 18-22 May 2015, Florence, Italy. *Corresponds to Chapter 1 of this thesis.*
- **Relationship-based top-k concept retrieval for ontology search:** *Anila Sahar Butt*, Armin Haller and Lexing Xie. In Proceedings of 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW'14), Pages 485-502, 24-28 November 2014, Linkoping, Sweden. *Corresponds to Chapter 6 of this thesis.*
- **Ontology search: An empirical evaluation:** *Anila Sahar Butt*, Armin Haller and Lexing Xie. In Proceedings of 13th International Semantic Web Conference (ISWC'14), Pages 130-147, 19-23 October 2014, Riva del Garda, Trentino, Italy. *Corresponds to Chapter 5 of this thesis.*
- **ActiveRaUL: A Web form-based User Interface to create and maintain RDF data:** *Anila Sahar Butt*, Armin Haller, Shepherd Liu and Lexing Xie. In Proceedings of the 2013th International Conference on Posters & Demonstrations (ISWC'13), Pages 117-120, 21-25 October 2013, Sydney, Australia.

Contents

Acknowledgments **vii**

Abstract **ix**

List of Publications **xi**

1 Introduction **1**
1.1 Problem Statement . . . 1
1.2 Application of Ontology Search . . . 4
1.3 Aim of Research . . . 6
1.4 Research Questions . . . 7
1.5 Research Methodology . . . 8
1.6 Contribution of this Work . . . 9
1.7 Thesis Outline . . . 12

2 Background **15**
2.1 Semantic Web . . . 15
2.1.1 Knowledge Representation in the Semantic Web . . . 15
2.1.1.1 Semantic Web data Formats . . . 16
2.1.1.2 Semantic Web Query Languages . . . 18
2.1.2 Ontologies . . . 19
2.1.3 Linked data . . . 22
2.2 RDF Stores . . . 23
2.2.1 RDF Databases . . . 23
2.2.1.1 Design goals of RDF databases: . . . 24
2.2.1.2 Types of RDF Databases: . . . 25
2.2.2 Need for RDF Stores . . . 25
2.2.2.1 RDF Databases vs. Existing database systems: . . . 25
2.2.2.2 RDF Databases vs. XML Stores: . . . 26
2.2.3 Storage Layouts and Access Mechanisms . . . 26
2.2.3.1 Native Stores . . . 26
2.2.3.2 Non-memory non-native Stores . . . 30
2.2.3.3 In-Memory Stores . . . 37
2.3 An overview of Semantic Web data retrieval process . . . 38
2.4 Summary . . . 41

3 Related Work **43**
3.1 Introduction . . . 43
3.2 Ontology Retrieval Techniques . . . 44
3.2.1 Ontology Libraries . . . 44
3.2.2 Ontology Search Engines . . . 48
3.2.3 Ontology Ranking Techniques . . . 49
3.3 Linked/RDF data Retrieval Techniques . . . 50
3.3.1 Efficiency oriented techniques . . . 50
3.3.2 Effectiveness oriented techniques . . . 51
3.4 Graph Structured data Retrieval Techniques . . . 55
3.5 Summary . . . 58

4 A Taxonomy of Semantic Web data Retrieval Techniques **59**
4.1 Introduction . . . 59
4.2 A Taxonomy of Semantic Web data Retrieval Techniques . . . 60
4.2.1 Retrieval aspects . . . 61
4.2.1.1 Retrieval Scope . . . 61
4.2.1.2 Query Model . . . 61
4.2.1.3 Result Type . . . 62
4.2.2 Storage and search approaches . . . 62
4.2.2.1 Data Acquisition . . . 62
4.2.2.2 Storage . . . 63
4.2.2.3 Indexing . . . 63
4.2.2.4 Query Match . . . 64
4.2.3 Ranking . . . 64
4.2.3.1 Ranking Scope . . . 64
4.2.3.2 Ranking Factor . . . 65
4.2.3.3 Ranking Domain . . . 66
4.2.4 Evaluation . . . 66
4.2.4.1 Efficiency . . . 66
4.2.4.2 Effectiveness . . . 67
4.2.4.3 Scalability . . . 68
4.2.5 Practical aspects . . . 68
4.2.5.1 Implementation . . . 68
4.2.5.2 Dataset . . . 69
4.2.5.3 User Interface . . . 69
4.3 Discussion . . . 69
4.4 Summary . . . 71

5 Evaluation Benchmark **75**
5.1 An introduction to CBRBench . . . 75
5.2 Baseline Ranking Models . . . 76
5.2.1 tf-idf . . . 77
5.2.2 BM25 . . . 78

5.2.3 Vector Space Model . . . 78
5.2.4 Class Match Measure . . . 79
5.2.5 PageRank . . . 80
5.2.6 Density Measure . . . 80
5.2.7 Semantic Similarity Measure . . . 81
5.2.8 Betweenness Measure . . . 81
5.3 Experiment setup . . . 82
5.3.1 Benchmark Ontology collection . . . 82
5.3.2 Benchmark query terms . . . 82
5.3.3 Establishing the ground truth . . . 84
5.3.4 Evaluation and Performance Measures . . . 85
Precision@*k*: . . . 85
Average Precision: . . . 85
Normalize Discounted Cumulative Gain (NDCG): . . . 85
5.4 Results . . . 86
5.4.1 Results Analysis . . . 89
5.5 Recommendations . . . 91
5.6 Summary . . . 93

6 Relationship-based Concept Retrieval 95
6.1 Introduction . . . 95
6.2 Concept Retrieval Framework . . . 96
6.2.1 Preliminaries . . . 96
6.2.1.1 Intra-Ontology Relationships . . . 96
6.2.1.2 Inter-Ontology Relationships . . . 97
6.2.1.3 Forward Link Concepts . . . 97
6.2.1.4 Back Link Concepts . . . 97
6.2.2 Overview of the framework . . . 97
6.2.2.1 Offline Ranking and Index construction . . . 97
6.2.2.2 Online Query Processing . . . 97
6.3 Offline Ranking and Index Construction . . . 98
6.3.1 DWRank: A Dual Walk based Ranking Model . . . 98
6.3.1.1 HubScore: The centrality of a concept within an ontology . . . 99
6.3.1.2 AuthorityScore: The authoritativeness of a concept . . . 100
6.3.1.3 DWRank Score . . . 101
6.3.2 Index Construction: An execution of DWRank . . . 102
6.3.2.1 ConHubIdx . . . 102
6.3.2.2 OntAuthIdx . . . 102
6.3.2.3 Inter-Ontology Relationships Extraction . . . 102
6.4 Online Query Processing . . . 103
6.4.1 Concept Retrieval Task . . . 104
6.4.2 Query Evaluation . . . 104
6.4.2.1 Candidate Result Set Selection . . . 105

6.4.2.2 Concept Relevance . . . 106
6.4.3 Filtering top-k results . . . 106
6.5 Experimental Evaluation . . . 108
6.5.1 Experimental Settings . . . 108
6.5.2 Experimental Results . . . 110
6.5.2.1 Effectiveness of DWRank . . . 110
6.5.2.2 Effectiveness of DWRank+Filter. . . . 110
6.5.2.3 Quality of HubScore - Measure of the centrality of a concept . . . 114
6.5.2.4 Effect of FindRel: Extraction of Implicit Inter-Ontology links . . . 114
6.6 Discussion . . . 117
6.7 Summary . . . 117

7 Learning Concept Ranking for Ontology Search 119
7.1 Introduction . . . 119
7.2 The Ranking Framework . . . 120
7.2.1 Learning to Rank . . . 120
7.2.1.1 Feature Set . . . 121
7.2.1.2 Training Data . . . 121
7.2.1.3 Metrics: . . . 124
7.2.1.4 LambdaMART . . . 124
7.2.2 Overview of the framework . . . 124
7.2.2.1 Index Construction and Learning Phase . . . 124
7.2.2.2 Query Processing Phase . . . 125
7.3 Evaluation Framework . . . 126
7.3.1 Experimental Settings . . . 126
7.3.2 Experimental Results . . . 126
7.3.2.1 Experiment-1: Effectiveness of Learning to Rank Approach . . . 126
7.3.2.2 Experiment-2: Effectiveness of Top-k Search . . . 127
7.4 Discussion . . . 128
7.5 Summary . . . 129

8 Ontology Recommendation for Structureless Queries 131
8.1 Introduction . . . 131
8.2 RecOn: Ontology Recommendation . . . 133
8.2.1 Preliminaries . . . 133
8.2.2 RecOn Workflow . . . 137
8.2.2.1 Query preprocessing . . . 137
8.2.2.2 Ontology Retrieval . . . 138
8.2.2.3 Ontology Evaluation . . . 138
8.2.2.4 Ontology Ranking . . . 139
8.3 Ontology Recommendation . . . 139

8.3.1 Ontology Retrieval . . . 139
8.3.2 Recommending k Ontologies . . . 140
8.3.2.1 Matching Cost . . . 140
Label matching cost. . . . 140
Structural matching cost. . . . 141
8.3.2.2 Informativeness . . . 142
8.3.2.3 Popularity . . . 143
8.3.2.4 Relevance Score . . . 144
8.4 Evaluation . . . 145
8.4.1 Experimental Setup . . . 145
8.4.1.1 DataSet . . . 145
8.4.1.2 Query selection . . . 146
8.4.1.3 Baseline: . . . 147
8.4.2 User Study . . . 149
8.4.2.1 Approach. . . . 149
8.4.2.2 Results . . . 150
8.4.3 Experiments . . . 152
8.4.4 Scalability Analysis . . . 153
8.4.5 Results Summary . . . 155
8.5 Summary . . . 156

9 Conclusion and Future Work **157**
9.1 Summary of our Contributions . . . 157
9.2 Future Directions . . . 160
9.3 Concluding Remarks . . . 162

List of Figures

1.1 The proposed research methodology 8

2.1 An Example of an RDF graph 16
2.2 An Example Ontology 20
2.3 An example diagram of relationship between Ontologies and data . . . 23
2.4 Structure and Example of NodeId Types 27
2.5 TDB Node storage architecture 28
2.6 Triple entry in SPO file 28
2.7 Quad entry in GSPO file 28
2.8 value-Id to value mapping in Sesame Native 29
2.9 value to value-Id mapping in Sesame Native 30
2.10 Triple in Sesame Native 30
2.11 SDB Layout in MySQL 31
2.12 SesameRDB Layout in MySQL 34
2.13 SesameM Storage Layout 38
2.14 Jena Subject Index Structure 38
2.15 Jena Predicate Index Structure 39
2.16 Jena Object Index Structure 39
2.17 Outline of the general Semantic Web retrieval process 40

4.1 The dimensions used to characterize Semantic Web retrieval techniques 60

5.1 Effectiveness of Ranking Model 86

6.1 Relationship-based top-k concept retrieval framework 98
6.2 Offline Index Construction 99
6.3 Online Query Processing 106
6.4 Effectiveness of Ranking Model (Tf-Idf = Term Frequency Inverse Document Frequency, BM25= BM25, VSM = Vector Space Model, CMM = Class Match Measure, DEM = Density Measure, SSM = Semantic Similarity Measure, PR = PageRank, BM = Betweenness Measure) 109
6.5 *AP@10* and *nDCG@10* for DWRank in comparison with the best value for any ranking model on sample queries 111
6.6 Filter Effectiveness 111
6.7 Comparison of Ranking Concepts 115

7.1 Generating training dataset for the proposed Learning Concept Ranking Framework . . . 122
7.2 Online Query Processing . . . 125
7.3 Effectiveness of Ranking Model(Tf-Idf = Term Frequency Inverse Document Frequency, BM25= BM25, VSM = Vector Space Model, CMM = Class Match Measure, DEM = Density Measure, SSM = Semantic Similarity Measure, PR = PageRank, BM = Betweenness Measure) . . . 128

8.1 An Example Ontology . . . 133
8.2 Query 'Q' and Query Graph 'QG' . . . 134
8.3 Match Patterns for 'Q' . . . 136
8.4 RecOn Workflow . . . 138
8.5 Effectiveness of $RecOn_{opt}$ vs. *AKTiveRank* for multi-term query strings 150
8.6 Effectiveness of $RecOn_{opt}$ vs. $RecOn_{ln}$ for multi-term query strings . . . 150
8.7 Effectiveness of query length on the $RecOn_{opt}$ performance . . . 151
8.8 Optional caption for list of figures . . . 151
8.9 Query match time for different length queries . . . 154
8.10 Query recommendation time for different length queries . . . 154
8.11 Optional caption for list of figures . . . 155

List of Tables

2.1 Prefixes table Design . . . 32
2.2 Nodes table Design . . . 32
2.3 Triples table Design . . . 32
2.4 Quads table Design . . . 32
2.5 Detail of Indices on SDB tables . . . 33
2.6 Property table description . . . 35
2.7 bnode_values/label_values/uri_values/long_uri_values/long_label_values /datetime_values/datatype_values/numeric_values/language_values tables description . . . 35
2.8 Hash_values table . . . 35
2.9 Namespace_prefixes table description . . . 36
2.10 Locked table description . . . 36
2.11 Detail of Indices on SesameRDB tables . . . 37

4.1 Categorization of Prominent SWR Techniques . . . 73
4.1 Categorization of Prominent SWR Techniques . . . 74

5.1 Notation used . . . 77
5.2 Query terms . . . 83
5.3 Ranking of *'Person'* in ground truth . . . 83
5.4 Precision @ 10 . . . 87
5.5 AP @ 10 . . . 87
5.6 DCG @ 10 . . . 88
5.7 NDCG @ 10 . . . 88
5.8 Overall average scores . . . 89

6.1 Top five reused ontologies based on explicit inter-ontology relationships 105
6.2 Top five reused ontologies based on implicit inter-ontology relationships 106
6.3 Intended Type Semantic Filter Performance in Relationship-based top-k Concept Retrieval Framework . . . 112
6.4 DWRank Effectiveness . . . 113
6.5 HubScore Quality: Centrality of Concepts . . . 113
6.6 Representative Ontologies . . . 114
6.7 HubScore Quality: Top-5 Concepts of Foaf Ontology . . . 116
6.8 Statistical Properties: Explicit vs. Implicit Inter-Ontology Links . . . 116

7.1 CBRBench Queries and their corresponding query IDs in training dataset 123

7.2 Learning concept ranking framework features and their corresponding feature IDs in training dataset . . . 123
7.3 Learning to Rank Concept Framework Effectiveness . . . 127

8.1 Functions with outputs . . . 134
8.2 Notations used throughout the chapter . . . 135
8.3 Query strings derived from benchmark query terms . . . 148
8.4 Comparison statistics of $RecOn_{opt}$ with baselines . . . 149
8.5 AP and NDCG for Single term queries . . . 152
8.6 Comparison statistics of $RecOn_{opt}$ with baselines . . . 153

Chapter 1

Introduction

In this chapter we provide an introduction to the work presented in this thesis. We describe the research problem in Section 1.1, the applications of the problem in Section 1.2, the aim of the research in Section 1.3, the research questions in Section 1.4, the methodology used to address the research problem in Section 1.5 and the contribution of this research in Section 1.6. We then provide an outline of this thesis in Section 1.7.

1.1 Problem Statement

The Semantic Web augments the World Wide Web (WWW) with semantics and provides a mechanism to define formal structures that are machine-readable and shareable. It enables data interoperability by creating a comprehensive and distributed data space for users and software agents to publish and access information from many different data sources [Noy and d'Aquin, 2012]. They can aggregate, integrate and use this information, regardless of its provenance and physical location. Semantic Web standards, such as the Resource Description Framework (RDF), make it possible for applications to perform more of the tedious work involved in finding, combining and acting upon information on the Web without human intervention.

Ontologies represent the essential technology that enables and facilitates interoperability at the syntactic and the semantic level. An ontology is a formal logic-based description of a vocabulary that allows one to conceptualize a domain of discourse [Staab and Studer, 2013]. The vocabulary is articulated using concept definitions and relationships among the defined concepts. As ontologies use formal logic, they can describe a domain unambiguously as long as the information is interpreted using the same logic. Moreover, the use of logic makes it possible to use software to infer implicit information in addition to what is explicitly stated [Noy et al., 2001]. However, only if an ontology, preferably a well-established and well-tested ontology, is reused and thus its conceptualization validated by others it becomes truly a shared conceptualization.

Reusability has long been recognised as a key attribute of ontologies. This is certainly true in principle, given that there are many generic concepts likely to appear, and thus be reused, in multiple applications [Katsumi and Grüninger, 2016]. The

process of reusing existing ontologies is cost-effective and can produce high-quality conceptualization both because it simplifies the process of building an ontology and because referring to an established ontological term in another domain of discourse creates an interlinked model of conceptualizations with strong formal semantics. Furthermore, ontology reuse significantly facilitates data interoperability as the use of the same ontology, by different data providers to describe their data, makes it possible for them to integrate their data much more easily. Moreover, It is almost always worth considering what someone else has done and checking if we can refine and extend existing sources for our particular domain and task. Reusing existing ontologies may be a requirement if our system needs to interact with other applications that have already committed to particular ontologies or controlled vocabularies.

With the development and recognition of the Semantic Web, more and more individuals, universities and industrial organizations have already started to publish their data in Semantic Web formats, such as the Resource Description Framework and the Web Ontology Language (OWL). The current explosion of Semantic Web data, for instance, the use of Schema.org,is accompanied by a rapid increase in the number of ontologies that are being developed and made available online [Bizer et al., 2009a]. In the last few years, several notable successes of ontology reuse have also been witnessed. For example, the well known FOAF ontology [Brickley and Miller, 2012], that describes people and their friends, is commonly used by softwares and applications to enrich the information described. Drupal makes use of FOAF ontology to automatically import profile information between Drupal-powered sites and from external foaf files. Similarly, SIOC uses FOAF to describe the creator of a post on blogs, forums or mailing lists. The GoodRelations ontology [Hepp, 2008] is used to describe products and offers of commercial companies. Moreover, some specific ontologies have achieved a high level of reuse as the communities in the domains agreed on foundational ontologies. For example, many biomedical researchers use the Gene Ontology [Ashburner et al., 2000] to annotate biomedical data. However, these examples of ontology reuse are still quite rare, and there is a need of enhancement of ontology reuse.

However, the process of reusing existing ontologies is related to significant costs and efforts, which may currently outweigh its benefits. First, as in other engineering disciplines, reusing some existing component implies costs to find, get familiar with, adapt and update the necessary modules in a new context. Second building a new ontology means partially translating between different representation schemes or performing scheme matching or both. A detailed discussion on ontology reuse and its use cases has been presented in [Katsumi and Grüninger, 2016; Bontas et al., 2005]. This thesis is motivated by the fact that it is hard to find the right ontology for a given use case.Despite the fact that ontologies provide a formal and unambiguous representation of domain conceptualization, it is rather expected to deal with different ontologies describing the same domain of knowledge, introducing heterogeneity to the conceptualization of the domain and difficulty in understanding or integration of information.

The growth of available ontologies in vertical domains such as bio-informatics,

e-commerce, and the internet-of-things highlights an increasing need of finding ontologies, and thus facilitating eventual data interoperability on the Semantic Web. The users must be able to find relevant ontologies quickly and easily. However, the process of choosing an ontology from an ontology collection is often a hard, manual and time-consuming task for researchers and other users. Two of the significant challenges faced by a user who needs to find an appropriate ontology for her application are: how to find the ontologies that deal with the required subject domain, and how to explore ontologies to find and determine which ones cover the domain sufficiently well for her to be able to use the ontology in her application.

Advocates of ontology reuse agree that one of the major barriers to the reuse of ontologies to design new ontologies or for the deployment of Linked Data is the difficulty that ontology and the data publishers have in determining which ontology to reuse to describe the semantics of their data [Vandenbussche and Vatant, 2014; Bontas et al., 2005; Noy and d'Aquin, 2012] The task of finding the right ontology for a given use case is often difficult for researchers. The researchers may not know the exact classes or properties and their positions in an ontology (ontological structure) they want, but require that the ontology contains a set of resources as its constituents. To mitigate the problem, a keyword-based query is commonly used for ontology search or recommendation. The problem here is that it is still hard to choose between ontologies that match to such a keyword query. Further, if there is no exact match for a given query string, a subset of the query may be used to find ontologies of interest. However, considering ontology matches for all subsets of the query terms results in a significant number of matches. Consequently, it is often too time-consuming for a user to explore the matched ontologies to find the most suitable one. Here comes the need of ontology retrieval frameworks that can find matched ontologies or its constituents for user-defined queries and then recommend the most suitable ontologies using ontology ranking models.

The growing number of online ontology libraries or search engines are beginning to address these challenges, enabling users to find and reuse ontologies, for example, [Alani et al., 2006; Noy et al., 2009; Noy and d'Aquin, 2012], has tackled the problem of finding and recommending ontologies. More recently, a dedicated ontology search engine has emerged [Vandenbussche and Vatant, 2014]. Some of the search engines (e.g. [Ding et al., 2004]) adopt document ranking algorithms to introduce ranking to their search results; most consider the popularity of terms in the ontology corpus. For this, they often use the PageRank algorithm as the ranking factor, which alone although effective in some cases (as shown in [Butt et al., 2014a]) hinders the visibility of newly emerging, but well-defined ontologies. Moreover, most of the ontology search systems retrieve ontological terms (concepts and relations), and only a few provide ontology search based on a keyword query. BioPortal [Noy et al., 2009] as the most prominent example provides functionality to find an ontology based on the text description. The approach used for ontology recommendation is domain specific (i.e., for biomedical ontologies) and is not applicable for all types of ontologies.

There is a need for an ontology retrieval framework that deals with domain independent ontology collections, uses effective metrics/models to evaluate matched

ontologies and then returns them in order of their usefulness to the user queries. In Section 1.2 we describe several example scenarios where finding a relevant ontology is required in real-world applications.

We formally define the problem of ontology recommendation as follows:

Definition 1.1. Keyword based Ontology Recommendation: Assume O_1, . . . ,O_n are the 'n' ontologies that belong to D_1, . . . , D_m the m subject domains. The user wishes to determine which of the ontologies or its terms (i.e., Concepts and Properties) best match to their domain of discourse based on q_1, . . . , q_k the k keyword query terms.

A viable ontology recommendation solution that can be used in real-world applications should address both the challenges of ontology reuse that have been identified in Section 1.1. There have been many different approaches proposed for ontology recommendation as recently surveyed in [Noy and d'Aquin, 2012; Butt et al., 2015]. As described in these studies, some attempts to address the problem of ontology recommendation fall short in providing a sound solution, either because they are not general or they are unable to provide a desirable ranking quality. Our research is an effort to address the problem of ontology retrieval and ranking. We propose a tool and an infrastructure that analyses and indexes ontologies to provide efficient and effective access to them for Semantic Web users. Given a set of keywords or terms representative of a domain of interest, our system recommends appropriate ontologies or concepts. Our research is relevant for both, researchers who need to select the best ontology for their domain and for Semantic Web developers who want to annotate their data with ontology selection capabilities.

1.2 Application of Ontology Search

Ontology retrieval and ranking for user defined queries with the aim to improve the quality of results is required in an increasing number of application areas including health care, government services, and business applications. Some of the applications of ontology search and ranking techniques are as follow:

1. **Ontology selection for data annotation:** Imagine a person or an application, which identifies entities of a certain type in Web pages, needs to annotate a news snippet *'The heroic mothers who raise children with special needs'*, and in search of an ontology containing the *Mother*, *Children*, *SpecialNeed* concepts. When queried for these terms, in the absence of an ontology search engine, it would be difficult for the person and almost impossible for the application to find the desired ontology. Therefore, a selection mechanism is expected to return an ontology that best covers them. Ontology libraries and search engines facilitate both persons or applications in finding the right ontology. The ontology may contain only a subset of the terms (partial coverage) as the user can extend the ontology according to her needs.

2. **Ontology reuse for building ontologies:** Building ontologies can be accomplished in one of the two approaches: it can start from scratch, or it can be built on the top of an existing ontology [Staab and Studer, 2013]. Let us suppose for a tourist management system a Semantic Web researcher or domain expert requires a tourist ontology to model famous tourist places, nearby accommodation facilities, restaurants and parking places. The researcher may design this ontology from scratch or can use an ontology recommender system to find the best ontologies that describe accommodation, restaurant and parking area concepts and use these concepts to build the new tourist ontology. Re-usability is the desired practice in this case both because the process of building an ontology from scratch is long and hard, and because the community needs to avoid the multiplication of several competing ontologies to represent the knowledge to realise the promise of the Semantic Web. The context of reuse has a significant influence on the requirements for the selection algorithm and should be taken into account when developing such algorithms.

3. **Ontology or linked data visualisation:** Ontologies and linked data are no longer exclusively used by Semantic Web experts but also by non-expert users in various domains. However, especially these casual users often have difficulties in understanding ontologies or linked data. Ontology or linked data visualisations help in this regard by assisting in the exploration, verification, and sense-making of such data. Many visualisation approaches visualise ontologies or linked data as graphs, which is a natural way to depict the structure of the data. However, the size of the linked data and the number of ontologies make it difficult to create understandable ontology or linked data graphs. Most of the ontology and linked data visualisation techniques summarise ontology collections or linked data by highlighting only important concepts or ontologies, or data that is generated using some famous concepts or ontologies in the data collection. Ontology ranking algorithms help ontology visualisation techniques to determine the most important ontologies or concepts and linked data visualisation techniques to choose the entities that belong to the important concepts.

4. **Understand a common conceptualization of a domain of discourse:** Ontologies describe the concepts and relationships in an area of knowledge using a logic-based language that enables automated reasoning. A user might be interested in understanding a particular domain of discourse and needs to understand the domain's core concepts and relationships. An ontology recommendation system shows the appropriate ontology for the domain of interest to the user.

1.3 Aim of Research

As discussed in Section 1.1, finding an appropriate ontology for a given use case to reuse is the core to realise the promise of Semantic Web (i.e. data interoperability and understanding). However, despite the many advantages of ontology retrieval framework, the topic is not well studied in the literature other than some domain specific ontology libraries, e.g. BioPortal for biomedical ontologies. Especially, there are only few ontology ranking models proposed or adopted for recommending the best results to the researchers and applications who look for matched ontologies for their use case. One of the reasons behind this is, there were a few ontologies before the explosion of Semantic Web [Guha, 2011]. Due to the difficulty of constructing ontologies, as well as the challenges of using ontologies in applications, researchers were hesitant in ontology development and reuse. Moreover, most existing ontologies are hard to reuse. The benefit of reusing existing ontologies are often unclear since the overhead of seeking and understanding existing ontologies by humans may be even greater than simply building an ontology from scratch.

With the advance of the Semantic Web, recently the number of ontologies has significantly increased [Butt et al., 2014a]. Since the use of ontologies in Semantic Web applications has improved performance, more people appreciate the advantage in using ontologies. The techniques for evaluating ontologies are necessary. Such techniques would not only be useful during the ontology engineering process, but they can also be useful to an end-user who needs to find the most suitable ontology among a set of technologies. Ontology retrieval techniques are particularly important in domains where large ontologies including tens of classes and properties are common.

Most ontology retrieval approaches that are proposed in the literature fall into two broad categories: the techniques for domain-specific ontology collections and the techniques for domain-independent ontology collections. The ontology retrieval techniques for domain-specific ontology collections generally perform considerably good; however, they use evaluation metrics or the methods to compute the evaluation metrics that are not applicable for domain-independent ontology collections. For instance, BiOSS [Martínez-Romero et al., 2014] uses the UMLS meta-thesaurus, and PubMed and BioPortal are specific to the biomedical domain only. On the other hand, the ontology retrieval techniques for the domain-independent ontology collections do not perform well for all queries and need improvements [Butt et al., 2014a]. Therefore, the primary aim of this research concentrates on ontology recommendation techniques that can be used for the domain-independent ontology collections.

1. **Extensive survey of ontology retrieval techniques:** Various techniques have been developed for Semantic Web data retrieval in general and ontology retrieval in specific over the past two decades. An extensive survey of such techniques is required to provide insights into the shortcomings of current techniques. Semantic Web data retrieval techniques involve many different dimensions and hence characterising existing techniques according to these di-

mensions for analysis, and the comparison is challenging, yet extremely useful to identify the research gaps and potential research directions in this domain.

2. **Ontology retrieval techniques for keyword queries:** Ontology or ontological terms' (i.e., concept and properties) retrieval techniques are used widely to recommend the most relevant terms, defined in ontologies, to the user queries. There have been some previous works; however, most of the existing works in their current form rank ontologies either based on the syntactic match of ontological terms to the query terms or their popularity in RDF datasets. The dataset based popularity of concepts or ontologies can be biased because an application mostly produces a dataset according to an ontology that is specifically designed for that application. These techniques do not analyse the semantic richness and the quality of a defined concept or ontology. There is a need for such retrieval techniques that can recommend concepts and ontologies according to their semantic richness and diversity that can enhance the information returned by an ontology retrieval framework.

3. **Evaluation framework for ontology retrieval techniques:** The evaluation of ontology retrieval techniques regarding the effectiveness of the ranking approach is important to allow the assessment and comparison of the performance of different solutions. A general framework with numerical and normalised measures for ontology recommendation will provide a baseline for comparison and analysis of ontology recommendation solutions. Developing such a framework is, therefore, important for ontology recommendation research.

1.4 Research Questions

On the basis of above discussion in Section 1.1 and 1.3 the main question this research aims to answer is: How can the internal structure of ontologies be utilised to increase the effectiveness of ontology retrieval systems?

To help answer this question, the following list of sub-questions should be answered:

- **RQ1:** What are the limitations of existing approaches for ontologies retrieval?
- **RQ2:** Do the document ranking models suffice for ontology ranking?
- **RQ3:** How to rank relevant resources and ontologies for keyword queries?
- **RQ4:** How to find the most relevant resources and ontologies that cover one or more resources users are interested in?
- **RQ5:** What are the inherent characteristics of ontologies that can be used to rank relevant resources and ontologies for keyword queries?

- **RQ6:** How to evaluate the newly emerging ontology libraries and search engines in comparison to existing ones?

1.5 Research Methodology

A research methodology, shown in Figure 1.1, adapted from [Kothari, 2004] is followed in conducting our research by applying the following steps:

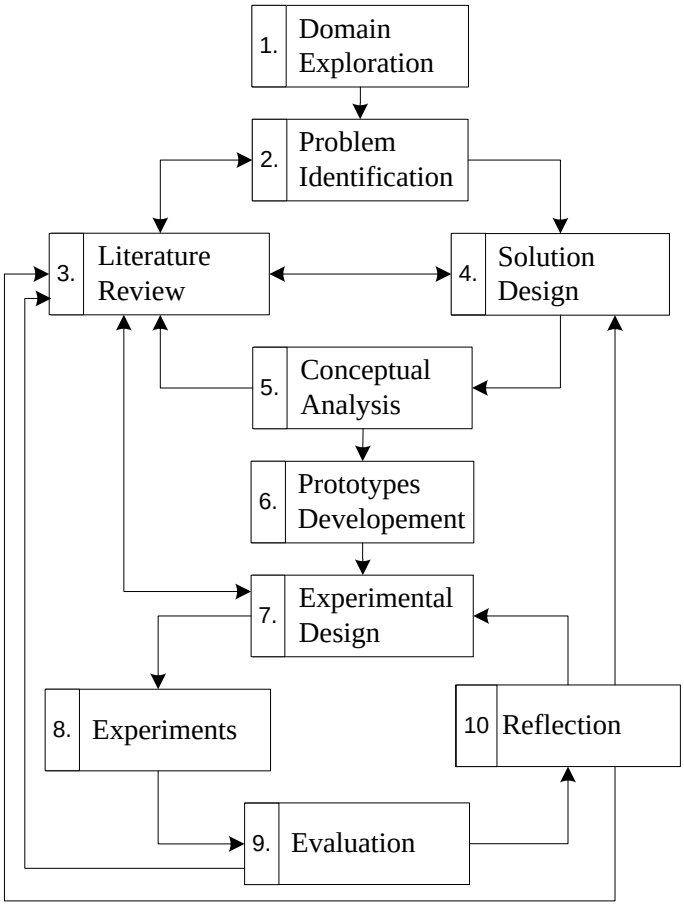


Figure 1.1: The proposed research methodology

1. **Domain exploration:** In this step a general understanding of the Semantic Web and the Ontology retrieval process was achieved by reading broadly in the literature which helped in recognising the research problem. A detailed literature is explored, from research repositories like Google scholar, related to ontology and other graph-based data retrieval processes; the recent and most cited work is considered as the related work for this thesis.

2. **Problem identification:** The research problems are identified based on the preliminary study that was conducted in the domain exploration step.
3. **Literature review:** In this step, an extensive investigation and examination of the literature were performed to understand the area of the identified problem better, i.e. ranking of graph-based data retrieval approaches including ontologies and RDF data, and review of current approaches. This step helped in defining the problem, identifying the aim of the research, and planning for possible solutions.
4. **Solution design:** Based on the problem definition and based on the review of the literature, possible solutions were proposed. New algorithms were designed to address the identified research problem.
5. **Conceptual analysis:** A theoretical analysis of the developed algorithms was conducted during this step about complexity, and estimation of the number of comparisons required between a query record and data sets' records.
6. **Prototypes development:** Prototypes of the proposed solutions were developed to be used in the experimental evaluation.
7. **Experimental design:** The experimental studies was designed, and data sets and evaluation measures were selected based on the literature review related to the experimental studies.
8. **Experiments:** The experimental studies are conducted.
9. **Evaluation:** The developed prototypes were experimentally evaluated using real-world ontology datasets to validate proposed algorithms.
10. **Reflection:** Finally, experimental results are reflected upon with regards to experimental design, dataset selection, and algorithm design.

1.6 Contribution of this Work

This thesis provides a detailed study of ontology retrieval techniques. Specifically, it proposes new ranking algorithms for concept and ontology retrieval addressing the research questions presented in Section 1.4. We group the contributions into three categories, which are (a) conceptual, (b) evaluative, and (c) methodological. The thesis mainly covers:

(a) **Conceptual:**

- **A taxonomy of Semantic Web data retrieval techniques** (Chapters 3 and 4): As discussed in Section 1.3, conducting an extensive survey of Semantic Web data retrieval techniques and more specifically ontology retrieval techniques with regard to different dimensions of ontology retrieval process is important to analyse the shortcomings in the current approaches.

We are the first to carry out such a large-scale survey in this domain. We present a taxonomy of Semantic Web data retrieval techniques that characterises existing such techniques along 16 dimensions in Chapter 4. These 16 dimensions are categorized into five main topics which are *retrieval aspects*, *storage and search*, *ranking*, *evaluation*, and *practical aspects*. We then characterise around 34 prominent Semantic Web data retrieval techniques that have been proposed in the literature in the last two decades (as surveyed in Chapter 3) along the proposed taxonomy. Further, we analyse the gaps that exist in existing techniques that will provide directions for future research. This contribution addresses the first research question **RQ1**, i.e., "What are the limitations of existing approaches for ontologies retrieval?"

(b) **Evaluative:**

- **An evaluation framework for ontology ranking models** (Chapter 5): One main shortcoming (identified in our survey) we address in this thesis is an evaluation framework for ontology ranking models. We propose a benchmark suite named CBRBench, for Canberra Ontology Ranking Benchmark, including an ontology collection, a set of queries and a ground truth established by human experts for evaluating ontology ranking algorithms. Also, we propose a methodology for resource ranking evaluation where we discuss many of the decision that needs to be made when designing a search evaluation framework for resources defined in ontologies. We implement eight ontology ranking algorithms and compare the ground truth derived through the human assessment of the results from each of the ranking algorithms. We calculated the precision at k, the mean average precision and the discounted cumulative gain of the ranking algorithms in comparison to a ground truth to determine the best model for the task of ranking resources or ontologies. Finally, a set of recommendations derived from an analysis of our experiment that we believe can significantly improve the performance of the ranking models is presented. The contribution addresses the two research questions **RQ6** and **RQ2** i.e., "How to evaluate the newly emerging ontology libraries and search engines in comparison to existing ones?" and "Do the document ranking models suffice for ontology ranking?"

(c) **Methodological:**

- **A concept retrieval framework** (Chapter 6): As discussed in Section 1.1, concept and ontology retrieval frameworks are required for data 'interpretability' and 'reuse' of Semantic Web data. There has been some previous work, for example, [Ding et al., 2005; Alani et al., 2006; Noy et al., 2009; Noy and d'Aquin, 2012], to tackle the problem of finding and ranking ontologies or concepts. More recently, also dedicated ontology search engines have emerged [Vandenbussche and Vatant, 2014], but the ranking algorithms they use are based only on document ranking algorithms.

Moreover, most of the ranking techniques in these ontology libraries and search engines only consider the popularity of terms in the ontology corpus, often using the PageRank [Page et al., 1998] algorithm, which although effective in some cases [Butt et al., 2014a] hinders the visibility of newly emerged well-defined ontologies. We propose a new ontology concept retrieval framework that uses a number of techniques to rate and rank each concept in an ontology based on how well it represents a given search term. The ranking in the framework is conducted in two phases. First, our offline ranking algorithm, DWRank, computes the centrality of a concept within an ontology based on its connectivity to other concepts within the ontology itself. Then, the authority of a concept is computed which depends on the number of relationships between ontologies and the weight of these relationships based on the authority of the source ontology. The assumption behind this is that ontologies that reuse and are reused by other ontologies are more authoritative than others. In a second, online query processing phase a candidate set for a *top-k* concept is selected from the offline ranked list of ontologies and then filtered based on two strategies, the diverse results semantics and the intended type semantics. The resulting list of *top-k* ranked concepts is then evaluated against a ground truth derived through a human evaluation published previously [Butt et al., 2014a]. Our evaluation shows that DWRank significantly outperforms the state-of-the-art ranking models on the task of ranking concepts in ontologies for the benchmark queries in the ontology collection. The contribution addresses the research question **RQ5** i.e. "What are the inherent characteristics of ontologies that can be used to rank relevant resources and ontologies for keyword queries?"

- **DWRank: Learning concept ranking for ontology search** (Chapter 7): Another important shortcoming we identified is that most of the ontology or concept ranking approaches consider one or more ranking metrics to get the best ordering of searched results [Alani et al., 2006; Tummarello et al., 2010; Hogan et al., 2011] and most of these approaches assign fixed weights to each metric while combining more than one evaluation metrics. However, [Butt et al., 2014a] shows that none of the commonly used evaluation metrics performs adequately. Moreover, for optimal performance of an algorithm, the metrics' weights need to be reset for each user query [Alani et al., 2006]. There can be two possible solutions to solve the problem i.e., the use of optimisation algorithms (dynamic programming) or machine learning approaches. We used a learning to rank approach to enhance the effectiveness of concept ranking models. The ranking metrics, i.e. *Text relevancy*, *Hub Score* and *Auth Score*, defined in Chapter 6 are extended and used to learn a ranking model that combines these measures in a more effective way. The effectiveness of the proposed approach is measured by comparing the ranking produced by the proposed approach with DWRank [Butt et al., 2014b]. Moreover, a comprehensive compari-

son of the proposed approach with state-of-the-art ranking models is also presented. The evaluation results show that the ranking produced by the proposed approach is more effective as compared to the baseline ranking models on CBRBench ontology collection and benchmark queries. The contribution addresses the research question **RQ3** i.e., "How to rank relevant resources and ontologies for keyword queries?"

- **RecOn: An ontology recommendation framework for structure-less queries** (Chapter 8): To address the research question **RQ4** i.e., "How to find the most relevant resources and ontologies that cover one or more resources users are interested in?", we propose `RecOn`, an Ontology Recommendation approach, an effort towards a dedicated ontology search engine that recommends relevant ontologies in response to a multi-term query string. Given a keyword query Q and a partial match approach, one might find many matches of Q in an ontology corpus. Thus, a user-friendly ontology search engine must address the following two questions: (1) how to determine which match is better, and (2) how to identify the top k matches? Our proposed ontology recommendation approach first finds the matched (relevant) ontology set to a query string; and then identifies the up to k most relevant ones. To identify the k most relevant ontologies for a query string, three measures are computed for each ontology: *matching cost* - the syntax and structural difference of the ontology from the query, *informativeness* - the information an ontology contains about the concepts that match the query string and *popularity* - the popularity of the ontology in the ontology corpus. We then find the relevance of an ontology to the query by formulating and solving the ontology recommendation as a linear model, referred to as $RecOn_{ln}$, and as an optimisation problem referred to as $RecOn_{opt}$. The aim is to find the ontologies that are as informative and popular as possible while incurring the least matching costs. The approach is evaluated on the CBRBench dataset [Butt et al., 2014a] against *AKTiveRank* by conducting a user study. The results of our user study show that $RecOn_{opt}$ and $RecOn_{ln}$ outperforms the state-of-the-art baseline algorithm *AKTiveRank*; and $RecOn_{opt}$ is efficient as well as effective as compared to $RecOn_{ln}$ on CBRBench ontology collection and sample queries designed in this work.

1.7 Thesis Outline

We begin by discussing the preliminaries of the Semantic Web in Chapter 2 and we survey existing Semantic Web data retrieval techniques in Chapter 3. In Chapter 4 we propose a taxonomy of Semantic Web data retrieval techniques, and characterise existing techniques along the proposed taxonomy to identify research directions. In Chapter 5 we present an evaluation framework for ontology ranking models that can be used for evaluation and comparison in the following chapters. We then propose a

relationship-based concept retrieval framework along with a dual walk based ranking model in Chapter 6 to address the effectiveness challenge, and convert the proposed ranking model into learning to rank technique by learning the feature weights using learning to rank algorithm in Chapter 7. In Chapter 8 we present efficient and effective ontology ranking algorithms based on evaluation metrics proposed in this chapter. Finally, we conclude the thesis by summarising our findings and discussing future research directions in Chapter 9.

Chapter 2

Background

This chapter summarises the background material that contributes to the understanding of fundamental concepts and techniques in the Semantic Web in general in Section 2.1, RDF stores in Section 2.2 and then describes the overall process involved in the Semantic Web data retrieval in Section 2.3.

2.1 Semantic Web

2.1.1 Knowledge Representation in the Semantic Web

Resource Description framework (RDF) is an underpinning language for information representation in the Semantic Web. It is a directed labelled graph-based data model to describe resources and their relationships in the Semantic Web as shown in an example in Figure 2.1. *Resources* are the things that are being described by the RDF expressions, and a resource could be an *instance* or a *concept* from a domain of discourse. *Predicates* are a special kind of resource that describes the relationships between other resources. Each resource, either is an instance, concept or predicate and is assigned a unique identifier that is called *Universal Resource Identifier* (URI).

An RDF graph is commonly interpreted as a set of triples (also known as RDF statements), where each triple contains a subject, predicate, and object in the form of <*subject, predicate, object*> (e.g. <person1, authorOf, paper1>). The subject is the source of an edge (i.e, 'person1'), the predicate is the edge itself (i.e, 'authorOf'), and the object is its target (i.e, 'paper1'). Resources that are subject in some triples can also appear as an object in some other triples. The subject and object do not restrict the view of the triples; an inverse predicate (e.g., hasAuthor) of the predicate (e.g., authorOf), can exchange the subject and object of a triple. The subject and predicate are always resources, in an RDF graph, but the object can either be a resource or a literal. A literal could be plain, such as a string, number, date, or an arbitrary sequence of characters; or it could be adorned with XML Schema datatype information. A literal could also be enhanced with a language tag to define the same literal in multiple languages. The example RDF graph, shown in Figure 2.1, has several instances: person1, paper1, and conference1, and concepts: Person, Publication, and Conference, both represented by ovals. There are some literal values drawn with

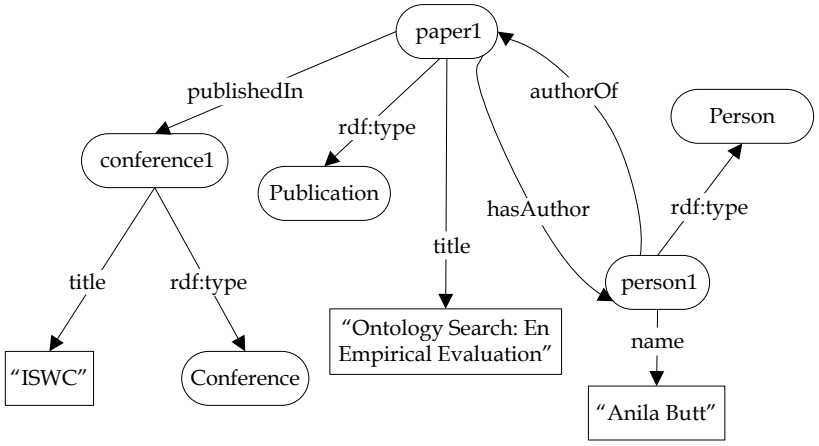


Figure 2.1: An Example of an RDF graph

rectangles, including *'Anila Butt'* and *'ISWC'*. The edges are predicates that connect concepts, instances, and literals.

RDF also defines a distinguished predicate, known as rdf:type, to define a type (i.e, class) of an instance. For example, the following triple, *<http://anu.edu/person1, rdf:type, http://www.example.com/publication#Person>*, states that *'Anila Butt'*, who is identified by the subject URI, is an instance of the *'Person'* class defined in this domain of discourse.

2.1.1.1 Semantic Web data Formats

Semantic Web data are directed labelled graphs as discussed previously; however, various formats are used for representing the data in text including RDF/XML, N-Triple, Turtle, and N-Quad. RDF/XML[1] represents data in the traditional XML format that is very verbose. N-Triples[2] and Turtle[3] have improved human readability for their design; and both formats are succinct compared to RDF/XML. N-Triples or Turtle make it possible for the users to write down RDF graphs by using triples. Then Semantic Web APIs, such as Sesame, Jena, and Virtuoso, are used for loading data in the Semantic Web formats that enable semantic queries over RDF graph. N-Quads[4] that is very similar to N-Triples and Turtle extends the option of indicating the context or source of a triple. For this purpose, an optional context field is added in the triple to show the context (known as provenance) of that triple. The example RDF graph shown in Figure 2.1 is presented in RDF/XML format in Listing 2.1 and in Turtle format in Listing 2.2.

[1]http://www.w3.org/TR/xmlschema-formal/
[2]http://www.w3.org/2001/sw/RDFCore/ntriples/
[3]http://www.w3.org/TR/turtle/
[4]http://sw.deri.org/2008/07/n-quads/

Listing 2.1: An example of representing Semantic Web data in RDF/XML Format

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:dc="http://purl.org/dc/terms/"
     xmlns:foaf="http://xmlns.com/foaf/0.1/"
     xmlns:pub="http://www.example.com/publication#">

    <pub:Conference rdf:about="http://anu.edu/conference1">
        <dc:title>ISWC</dc:title>
    </pub:Conference>

    <pub:Publication rdf:about="http://anu.edu/paper1">
        <dc:title>Ontology Search: An Empirical Evaluation<dc:title>
        <pub:publishedIn rdf:resource="http://anu.edu/conference1"/>
        <pub:hasAuthor rdf:resource="http://anu.edu/person1"/>
    </pub:Publication>

    <pub:Person rdf:about="http://anu.edu/person1">
        <foaf:name>Anila Butt</foaf:name>
        <pub:authorOf rdf:resource="http://anu.edu/paper1"/>
    </pub:Person>

</rdf:RDF>
```

Listing 2.2: An example of representing Semantic Web data in Turtle Format

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/terms/> .
@prefix pub: <http://www.example.com/publication#> .

<http://anu.edu/conference1> a pub:Conference;
                             dc:title "ISWC".

<http://anu.edu/paper1> pub:hasAuthor <http://anu.edu/person1>;
                        pub:publishedIn <http://anu.edu/conference1>;
                        a pub:Paper;
                        dc:title "Ontology Search: An Empirical Evaluation".

<http://anu.edu/person1> pub:authorOf <http://anu.edu/paper1>;
                         a pub:Person;
                         foaf:name "Anila Butt".
```

2.1.1.2 Semantic Web Query Languages

Special purpose Semantic Web query languages have been developed to retrieve and manipulate data stored in Semantic Web formats, and a comparison of such languages is presented in [Haase et al., 2004a]. Semantic Web query languages are more complex than SQL because of the underlying RDF graph data model. Specifically, while a relational query executes over one or more tables each containing tuples with the same structure, an RDF query executes over an RDF container that may contain resources of different types each with different properties. Moreover, values of properties, rather than being mere data, can be resources themselves. Given an RDF graph D, a Semantic Web query consists of one or more pattern/s that is matched against D, and the values obtained from this matching are processed to retrieve the final answer.

Until now several designs and implementations of Semantic Web query languages have been proposed including SeRQL[5] and RDQL[6]. However, SPARQL[7] is the W3C recommended query language for Semantic Web data. An example SPARQL query is shown in Listing 2.3. The query finds all publications of *'Anila Butt'* in ascending order of publication title from the RDF graph shown in Figure 2.1. In general, a SPARQL query has three constituents: Pattern matching, a solution modifier and the output of the query, where the *SELECT* clause defines the required *output*. Everything inside the *WHERE* clause is *pattern matching*. The pattern matching part includes several interesting features of pattern matching of graphs, e.g. optional parts, union of patterns, nesting, filtering (or restricting) values of possible matching and the possibility of choosing the data source to be matched by a pattern. The *solution modifiers* allow to modify the pattern matching values by applying classical operators like distinct, order, limit, and offset.

Listing 2.3: An example SPARQL Query

```
1  #List all publications of "Anila Butt" in an ascending order of publication
       title.
2
3  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
5  @prefix dc: <http://purl.org/dc/terms/> .
6  @prefix pub: <http://www.example.com/publication#> .
7
8  SELECT ?paperTitle
9  WHERE {
10     ?paper pub:hasAuthor ?author.
11     ?paper dc:title ?paperTitle.
12     ?author foaf:name "Anila Butt".
13  } ORDER BY ?paperTitle
```

[5]http://www.csee.umbc.edu/courses/graduate/691/spring14/01/examples/sesame/openrdf-sesame-2.6.10/docs/users/ch09.html

[6]http://www.w3.org/Submission/RDQL/

[7]http://www.w3.org/TR/rdf-sparql-query/

2.1.2 Ontologies

The goal of the Semantic Web is to automate machine processing of web documents by making their meanings explicit [Staab and Studer, 2013]. As discussed previously, RDF is a graph-based data model for describing resources and their relationships in the Web; however, it does not allow users to add any semantics to the data. RDF is only a *Data modelling language*. Semantic Web researchers introduced *Domain modelling languages* that enable explicit semantics to the content structuring aspects of the Extensible Markup Language (XML)[8]. Such language allows users to create ontologies (i.e. RDF Vocabularies) [Gruber, 1995], which specify standard terms and machine-readable definitions. Information resources (such as web pages and databases) then commit to one or more ontologies, stating which sets of definitions are applicable. Minimally, an ontology is an explicit and formal specification of conceptualization, formally describing a domain of discourse [Staab and Studer, 2013]. An ontology consists of a set of terms (classes or concepts) and their relationships (class hierarchies and predicates). For example, a publication ontology is shown in Figure 2.2 (that models publications, conference papers, journal papers, authors, and person, etc.) might state that both *Conference Paper* and *Journal Paper* are subclasses of the *Publication* class and that they are disjoint, i.e., no single publication can be a *Conference Paper* and *Journal Paper* at the same time. Moreover, each conference paper (resp. journal paper) has a *bookTitle* (resp. *journal name*) in addition to a title and one or more *authors*. These definitions describe some of the meaning of the terms at the schema level. The use of ontologies make intended meaning of assertions unambiguous by adopting formal logic, and therefore, it avoids the ambiguities of natural language.

There are two major *domain modelling languages* available on the Semantic Web. *RDF Schema (RDFS)* is a weak expressive ontology language and is used to provide some basic semantics for the classes and properties, and it augments the RDF by adding some basic constructs. These constructs include classes, properties, class hierarchies, property hierarchies, domain, and range. The vocabulary used to model these extended constructs are *rdfs:Class*, *rdfs:Property*, *rdfs:subClassOf*, *rdfs:subPropertyOf*, *rdfs:domain* and *rdfs:range* respectively. These constructs allow statements about something's type by making new statements from existing statements such as triple <http://anu.edu/anila_butt, rdf:type, http://www.example.com/publication#Author> depicts that *'Anila Butt is an author'*, and with an additional statement that <http://www.example.com/publication#Author, rdfs:subClassOf, http://www.example.com/ publication#Person>, machines can infer that *'Anila Butt is a person'*. Moreover, domain and range of a predicate describe the classes of things that can be declared as a subject or an object of a predicate, e.g., *'authorOf'* can have an instance of class *'Person'* as a subject and an instance of class *'Publication'* as an object in an RDF statement or triple. RDFS defines the semantics of the application domain; however, there are some characteristics (e.g., disjointness and the boolean combination of classes, cardinality restrictions and special characteristics of predicates) for ontologies on the

[8]http://www.w3.org/TR/xml/

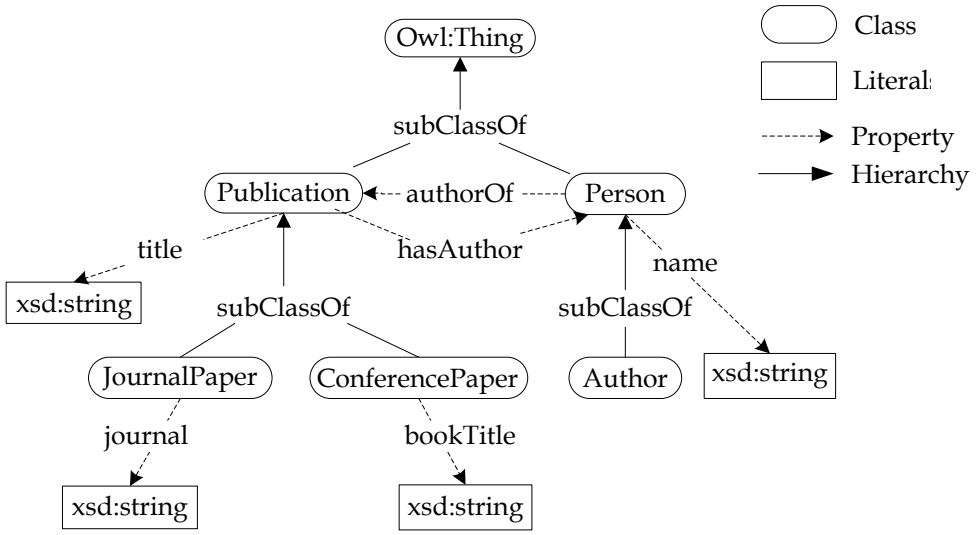


Figure 2.2: An Example Ontology

web which would require much more expressiveness than that provided by RDF Schema [Antoniou and Van Harmelen, 2004].

Web Ontology Language (OWL2) is another domain modelling language designed specifically for the Web that is compatible with XML, as well as other W3C standards. On the one hand, OWL2 adds significant expressivity for describing the semantics of ontologies; on the other hand, syntactically, an OWL2 ontology is a valid RDF document and a valid XML document. This enables ontologies and documents written in OWL2 to be processed by the wide range of XML and RDF tools that are already available, such as Jena [Carroll et al., 2004]. OWL2 extends the RDF schema and adds some language primitives that support more expressiveness by extending constructs including OWL classes, properties (object property and data type properties), property restrictions, equality and inequality of classes, properties characteristics and restrictions, cardinalities, boolean combinations, enumerations, and versioning information. Although OWL2 provides a higher level of expressiveness than RDFS, the richer the language is, the more inefficient reasoning support becomes. Depending upon a tradeoff between language expressiveness and efficient reasoning, OWL2 has three variants i.e. OWL2 EL, OWL2 QL, and OWL2 RL.

As a concrete example, Figure 2.2 demonstrates a toy ontology that defines some classes and properties. OWL2 syntax for the ontology is shown in Listing 2.4. At the beginning of this ontology syntax, there is a definition of two object properties (i.e., relationship between two resources), called *'authorOf'* and *'hasAuthor'*, followed by the definition of datatype properties (i.e., relationship between a resource and a literal), called *'title'*, *'journal'*, *'bookTitle'* and *'name'*. Note that the *'authorOf'* is an inverse property of *'hasAuthor'* property. *'hasAuthor'* property connects *'Publication'* to *'Author'*, specified by its domain and range. Moreover, classes are defined (line 25-

Listing 2.4: An example of OWL format

```
<rdf:RDF xmlns="http://www.example.com/publication#"
     xml:base="http://www.example.com/publication"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:terms="http://purl.org/dc/terms/"
     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#">

    <owl:ObjectProperty rdf:about="#authorOf">
        <owl:inverseOf rdf:resource="#hasAuthor"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="#hasAuthor">
        <rdfs:domain rdf:resource="#Publication"/>
        <rdfs:range rdf:resource="#Author"/>
    </owl:ObjectProperty>

    <owl:DatatypeProperty rdf:about="&terms;title">
        <rdfs:domain rdf:resource="#Publication"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:about="#journal"/>
    <owl:DatatypeProperty rdf:about="#bookTitle"/>
    <owl:DatatypeProperty rdf:about="#name"/>

    <owl:Class rdf:about="#Author">
        <rdfs:subClassOf rdf:resource="#Person"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#authorOf"/>
                <owl:onClass rdf:resource="#Publication"/>
                <owl:minQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger"
                    >1</owl:minQualifiedCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:about="#ConferencePaper">
        <rdfs:subClassOf rdf:resource="#Publication"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&terms;title"/>
                <owl:someValuesFrom rdf:resource="&xsd;string"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:about="#JournalPaper">
        <rdfs:subClassOf rdf:resource="#Publication"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#journal"/>
                <owl:someValuesFrom rdf:resource="&xsd;string"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:about="#Person">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#name"/>
                <owl:someValuesFrom rdf:resource="&xsd;string"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:about="#Publication">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasAuthor"/>
                <owl:allValuesFrom rdf:resource="#Author"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
</rdf:RDF>
```

69) that have a little more complex definition as compared to properties. For better understanding, lines 25 -34 simply say that an author is the one who is a person and is authorOf at least one publication.

2.1.3 Linked data

The core of the Semantic Web is the Web that has some ontologies, modelling different or similar domains of discourse, and data generated according to these ontologies. These ontologies and data sources can be source-specific - being constructed or generated for a particular application without considering future integration, or generic - are extendable to aid future integrations. When two or more data sources or ontologies bind to the same ontology, explicitly by using the owl:imports predicate or implicitly by just using the ontology's namespace, then the similar meaning is intended for a resource from that ontology. This idea is decentralised in that any data source or ontology can bind to any ontology, and any data source can create a new ontology.

In the Semantic Web, *Linked Data* enables the Web to connect data that is related but previously disconnected or lowers the barriers to connecting data currently linked using some other technologies. For example, in Figure 2.3, the dashed lines between data sources D4 and D6 represent the co-reference relationships between their instances.

> "Linked data describes a method of publishing structured data so that it can be interlinked and become more useful. It builds upon standard Web technologies such as HTTP and URIs, but rather than using them to serve web pages for human readers, it extends them to share information in a way that can be read automatically by computers. This enables data from different sources to be connected and queried." - [Bizer et al., 2009a]

According to the above quote, Linked Data does not only expose data using Web technologies, or not merely an elegant way to solve interoperability issues, but it helps in achieving the fundamental goal of the Semantic Web, *a transition from a web of documents to a web of data*. For that Tim Berners-Lee, the inventor of the idea of the Semantic Web specifies four basic principles about Linked Data[9]. These principles are: (1) use URIs as names for things, (2) use HTTP URIs so that people can look up those names, (3) when someone looks up a URI, provide useful information, using the standards like RDF or SPARQL, and (4) include links to other URIs, so that users can discover more things. The first two principles enable entities or things to be identified and searched on the Web. The last two principles are worth mentioning as they allow the Web and the Semantic Web for storing information in a distributed manner at multiple places. Moreover, the links between the identifiers that are placed on different data sources allows users to navigate and obtain more information.

Considering the RDF graph shown in Figure 2.1 as an example. If *'Anila Butt'* has published in two such conferences that released their publication data as Linked

[9]http://www.w3.org/DesignIssues/LinkedData.html

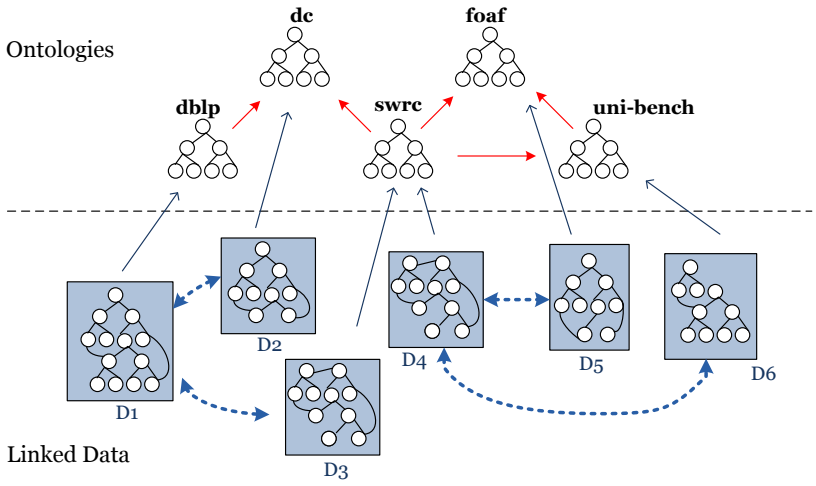


Figure 2.3: An example diagram of relationship between Ontologies and data

Data. The software can integrate the information, using the biographical information of the author available on DBpedia, with a simple query, as if all the data was in a single database. Currently, hundreds of different datasets exist on the Web including Wikidata [Vrandečić and Krötzsch, 2014] and DBpedia [Bizer et al., 2009b], published according to the Linked Data principles with an open license that can be used and republish without restrictions. When combined with Linked Data, we finally have *Linked Open Data*, where data is publicly available and well interlinked to each other.

2.2 RDF Stores

RDF stores, also known as Triple or Quad Stores, are the databases of the Semantic Web world, designed to hold massive numbers of triples in such a manner that the information they encode can be simply retrieved. This section is further divided into two sections. Section 2.2.1 presents a brief introduction of RDF stores and section 2.2.2 gives an overview of the role of the RDF stores even in the presence of exiting XML stores and/or database systems i.e. relational DBMS, object oriented DBMS, and object relational DBMS.

2.2.1 RDF Databases

RDF databases are Semantic databases where semantic data can be conveniently stored, operated upon and retrieved [Owens, 2009]. A triple store can be defined as "A system to provide a mechanism for persistent storage and access of Semantic Web graphs." Its main functions include storing, reasoning and querying Semantic Web data. They employ index structures, algorithms for buffering, join, concurrency control for optimal query processing and reasoning. An intelligent query optimizer

in a triple store strives to save resources in terms of time and memory space in query processing and reasoning.

2.2.1.1 Design goals of RDF databases:

Some of the eminent design goals of RDF databases are:

- **Scalability:** Resources are described on the Semantic Web in terms of triples. A resource may need many triples for its perfect description. It is therefore necessary for RDF Databases to deal with a large number of triples in an elegant manner.

- **Dynamism and Network Distribution:** Data on the Semantic Web is dynamic as it belongs to different sources because of network distribution. RDF Databases may be used as a server or a client to handle timeouts, network failure, bandwidth use and deal with denial-of-services (DoS). RDF Databases should be able to manage the network resources in both when a server or a client and deal with dynamic data in a graceful way.

- **Unpredictability:** Semantic data is highly unpredictable in its nature. A Large number of triples, dissimilar terms used to describe resources, rate of triples exchange over the network and effect of network provide a high degree of unpredictability to the data. RDF Databases need to handle this unpredictable nature of data in an efficient and accurate manner.

- **Provenance:** As described earlier, the Semantic Web data comes across different sources that may necessitate keeping track of original location or context of the data. This part of information is called provenance. RDF Databases may need to store the context, along with the original information.

- **Data Processing:** Data in the RDF Databases need to be processed . This necessitates RDF Databases to provide some mechanism for accessing the RDF graphs, identifying triples, storing triples in data stores, merging data from multiple sources into single store, and querying as well as administering the data stores. These operations are performed by applications many times thus require RDF Databases to provide lightweight, fast, easy to use and understandable APIs to carry out these tasks. Many of the existing Semantic Web applications interact with human. So, they must perform as fast and accurate as possible in order to provide shield against frustration. Interactive level performance is one of the key requirements of these stores for human friendliness.

- **Reasoning:** A final clear issue is support for inference. RDF Databases support different degree of reasoning (RDF/RDFS/OWL) while RDFS support is common. Currently only a very few stores support OWL features, and they do not provide the performance measures while using these features whereas simpler systems do.

2.2.1.2 Types of RDF Databases:

Different RDF Databases have different architectures thus result in varying performance levels. Based on their storage structure and medium, RDF Databases can be divided into three broad categories, In-memory, Native, Non-native Non-memory.

- In Memory Stores: Triples are stored in main memory e.g. storing an RDF graph using BRAHM [Janik and Kochut, 2005].
- Native Store: Persistent storage systems that are disk resident with their own implementation of databases e.g. Virtuoso [Erling and Mikhailov, 2009] and AllegroGraph [Aasman, 2006].
- Non-memory non-native Stores: Non-memory non-native stores are disk resident and employ the existing database management systems such as Microsoft SQL, MySQL, and Oracle for storing triples. 3Store [36] is an example of this type of RDF Databases.

Hybrid of three classes is also available, for example Jena [McBride, 2001] and Sesame [Broekstra et al., 2002]. Triple stores have their own query languages to query store's data. List of existing large triple stores include BigOWLIM, Bigdata(R), Garlik, 4store ,YARS2, Virtuoso, Jena TDB, AllegroGraph, Jena SDB, Mulgara, RDF gateway, Jena with PostgreSQL, Kowari, 3store with MySQL, Sesame, TopQuadrant.

2.2.2 Need for RDF Stores

RDF is characterized by a property centric, extremely flexible and dynamic data model. Resources can acquire properties and types at any time, regardless of the type of the resource or property. This flexibility makes RDF an attractive technology for the specification and exchange of arbitrary metadata. The challenge is thus how to provide persistent storage for the new RDF data model in an efficient and flexible manner.

2.2.2.1 RDF Databases vs. Existing database systems:

Purpose of the RDF Databases is somewhat similar to the existing database systems i.e. management of stored data. RDF documents' storage necessitates special type of data stores because of two fundamental differences between RDF graph model and other data models e.g. relational data model [Codd, 1970] and object data model [Atkinson et al., 1989] that demand some special kind of data stores to manage RDF data. These two differences are:

1. Unpredictable structure of the data stored in RDF graph model
2. Unpredictable query patterns over this data in Semantic Web

All existing database systems require that structure of data (i.e. schema) must be defined before inserting that data [Date, 2006]. Predefined structure of data helps

in data integrity by constraining the incorrect data to be used by any organization or application. However in Semantic Web, where the interoperation between heterogeneous data sources is permissible, structure of data is unknown and changes continuously. Existing database systems are unable handle unstructured data. This gives rise to a data storage system that does not need any prior definition of the structure of data. Existing database management systems are used by known set of applications. Such databases can be optimized on the basis of metadata i.e. indexes and estimated statistical knowledge [Date, 2006] for most anticipated query patterns for these applications. Access for all other patterns is comparatively slower than these anticipated patterns. But RDF data can be accessed and manipulated by any node on the Semantic Web that requires RDF data stores to handle queries of unpredictable patterns. Both the reasons concluded above necessitate the proposal of some new storage systems that can better handle the complexity of RDF data and query.

2.2.2.2 RDF Databases vs. XML Stores:

One approach for RDF storage might be to map the RDF data to XML and simply leverage prior work on the efficient storage of XML. However, the standard RDF/XML mapping is unsuitable for this since multiple XML serializations are possible for the same RDF graph, making retrieval complex [Haase et al., 2004b]. Non-standard RDF-to-XML mappings are possible, and have been used in some implementations. However the simpler mappings are unable to support advanced features of RDF, such as the ability of RDF to treat both properties and statements as resources, which allows metadata describing these elements to be incorporated seamlessly into the data model and queried in an integrated fashion.

2.2.3 Storage Layouts and Access Mechanisms

2.2.3.1 Native Stores

Native stores provide persistent storage for Semantic Web data. These databases create disk based files to store Semantic Web data. Native stores implement different data structures, a detail study of few triple stores is provided here.

1. **Jena TDB:** stores RDF triples in a directory on the disk in filing system. Whenever a TDB store is created, it creates some files for the storage and retrieval of triples, that can be broadly divided into three categories that are Nodes, Prefixes, Triples and Quads.

    (a) **Nodes:** TDB files 'nodes' and 'node2id' provide two types of mappings from node to nodeId and from nodeId to node. The 'Node to NodeId mapping' is used during data loading and when converting constant terms in queries from their Jena Node representation to the TDB-specific internal ids. The 'NodeId to Node mapping' is used to turn query results expressed as TDB nodeIds into the Jena node representation and also during

query processing when filters are applied if the whole node representation is needed for testing e.g. regex. A nodeId can be of two types in Jena TDB

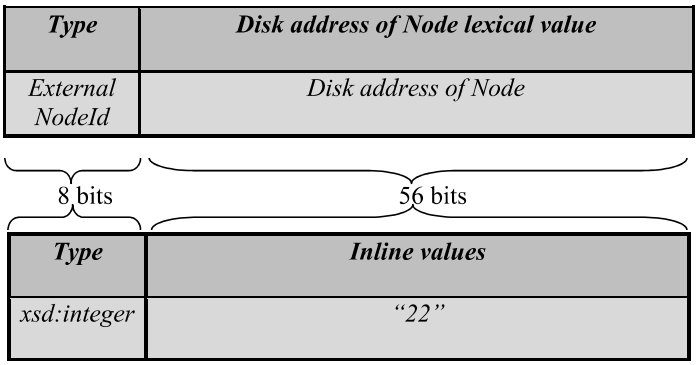


Figure 2.4: Structure and Example of NodeId Types

i.e. External nodeId or the Value space as shown in Figure 2.4. First byte of the nodeId stores the type of the nodeId. If type is external nodeId then next 7 bytes contain the physical address of the node as describe above. If the type of nodeId is value space then the values of data types, which are considered in value space, are stored as a part of nodeId in lower 7 bytes. Data types that are considered in value space are xsd:decimal, xsd:integer, xsd:dateTime, xsd:date and xsd:Boolean. The node file stores the actual Jena node representation. The node to nodeId mapping is based on the hash of the lexical value of the node and is stored in node2id file that is implemented as a B+ tree. The size of an entry in this file is of 24 bytes. The first 16 bytes are the hash value of node and next 8 bytes are the disk address of the node lexical value (except for the inline values) in the node file. Whenever a node is asserted into a TDB store MD5 hash of the node is computed and enters into the first 16 bytes. Then a unique id is assigned against this hash value, this id represents the physical address in node file. Actual lexical value of that node is stored at the address which is represented by the nodeId. The storage process of node is represented in the Figure 2.5.

(b) **Prefixes:** This category contains three files that are prefixes, prefix2id and prefixidx. These provide supports for TDB Prefix Mappings. Just like nodes and node2id, prefixes and prefix2id provide two types of mapping for prefixes and prefixIds. Prefixidx is another implementation of B+ tree that is ordered on GPU (Graph, Prefix, URI).

(c) **Triple and Quads:** Remaining files are categorized under this category. These are SPO, POS, OSP, GSPO, GPOS, GOSP, SPOG, POSG, and OSPG. There is no distinguishing triple file and then indexes on this file. SPO, POS and OSP are triple index files that have B+ tree implementation. These are populated when no provenance information is stored about the

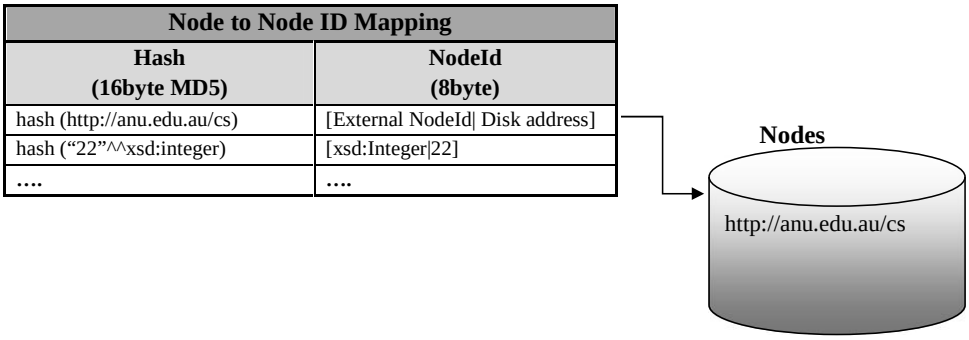


Figure 2.5: TDB Node storage architecture

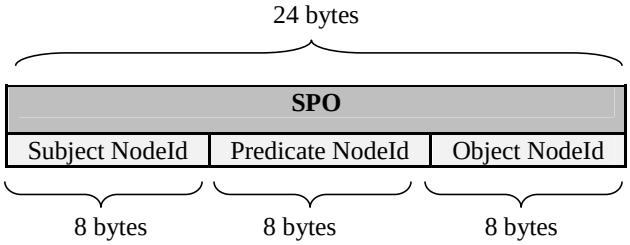


Figure 2.6: Triple entry in SPO file

triples. Each entry of these files is of 24 bytes and has all the information about a triple. Triples in these files are represented as a combination of three nodeIds in different orders, one for subject second for predicate and third for object. Name of each file represents the order of triple in terms of subject, predicate and object as shown in Figure 2.6 Whenever a triple is

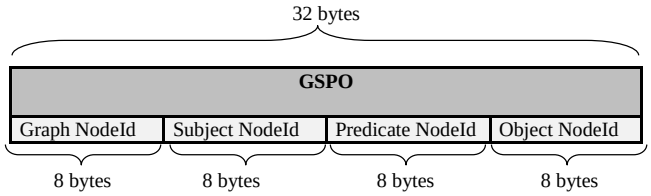


Figure 2.7: Quad entry in GSPO file

asserted into Jena TDB store three entries are made in three different files, one entry in each of SPO, POS and OSP files. Quads index files are used to represent the named graphs. Default storage of these files in Jena TDB is B+ tree. These are populated when provenance information is stored about the triples. Each entry in quad index files is of 32 byte representing subject, predicate, object and graph for a triple as shown in Figure 2.7. Whenever a quad is asserted into Jena TDB store six entries are made in six quad index files, one entry in each file.

2. **Sesame-Native:** (SesameN) stores triples permanently on disk for inferencing and querying of Semantic Web data. SesameN schema in the disk directory can broadly be divided into three logical blocks. (1) Namespaces (2) Values (3) Triples.

   (a) **Values:** This logical part is composed of three disk based files that are named as value.dat, value.id and value.hash. These three files provide two types of mapping i.e. "value to value-Id" and "value-Id to value". The "value to value-Id mapping" is used during data loading and when converting constant terms in queries from their node representation to the Sesame specific internal value-Ids. The "value-Id to value" is used to turn query results expressed as Sesame value-Id into the RDF node representation and also during query processing when filters are applied if the whole node representation is needed for testing e.g. regex. The

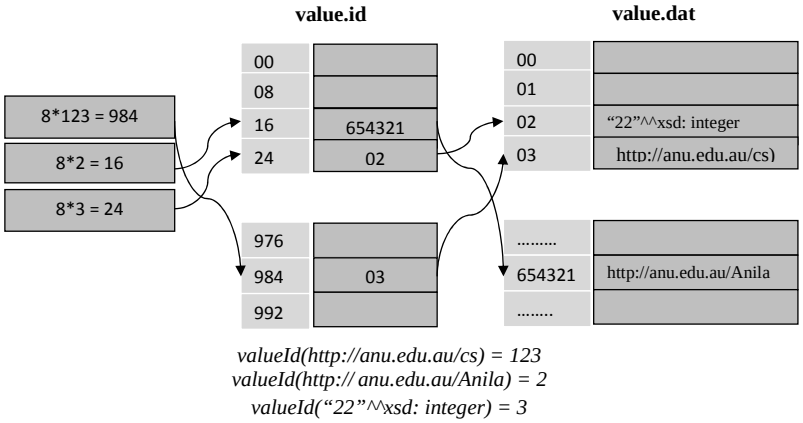


Figure 2.8: value-Id to value mapping in Sesame Native

   actual values of URIs, blank nodes and literals are stored sequentially in the values.dat file. Value-Id to value mapping is maintained in value.id file. It is a B-tree disk based implemented file. The values.dat offset for value X is stored in values.id at offset 8 * X, where X is a positive 32 bit integer and offsets are 64 bit longs. So, to look up the lexical value for id 123, the native store fetches the long stored at offset 8 * 123 = 984 from values.id. The value of this long is, for example, 654321. The lexical value can then be read from values.dat at offset 654321 as shown in Figure 2.8. The value to value-Id mapping is based on the value.hash file that is a disk based hash table. It stores value identifiers using a hash code derived from the actual value. Hash of any RDF node's lexical value (i.e. resource, literal or blank node) returns the physical address of the value-Id for that node, within the address space of value.hash as shown in Figure 2.9.

   (b) **Triples:** Triple-sopc, triple-posc and triple-cosp fall under this category. These are on-disk indexes to speed up querying. These uses B-trees for

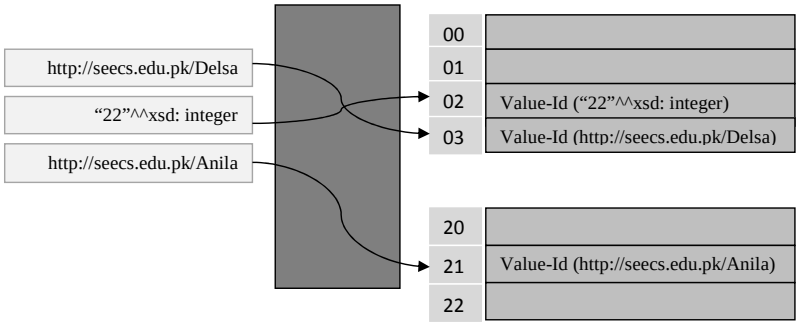


Figure 2.9: value to value-Id mapping in Sesame Native

indexing statements, where the index key consists of four fields: subject (s), predicate (p), object (o) and context (c). These file only store identifiers (integer ids) instead of actual URIs, blank nodes and literals as shown in Figure 2.10. The order in which each of these fields is used in the key determines the usability of an index on a specify statement query pattern. Searching statements with a specific subject in an index that has the subject as the first field is significantly faster than searching these same statements in an index where the subject field is second or third. In the worst case, the 'wrong' statement pattern will result in a sequential scan over the entire set of statements. By default, the native repository only uses two indexes, one

SOPC			
Subject valueId	Object valueId	Predicate valueId	Context valueId

Figure 2.10: Triple in Sesame Native

with a subject-predicate-object-context (spoc) key pattern and one with a predicate-object-subject-context (posc) key pattern. However, it is possible to define more or other indexes for the native repository, using the Triple indexes parameter. This can be used to optimize performance for query patterns that occur frequently. Creating more indexes potentially speeds up querying, but also adds overhead for maintaining the indexes. Also, every added index takes up additional disk space.

(c) **Namespaces:** A single file contains the namespaces of the dataset.

2.2.3.2 Non-memory non-native Stores

Non-memory non-native stores also provide persistent storage for Semantic Web data. They use the storage and querying techniques provided by existing RDBMS. Detailed study of different storage layouts deployed by Semantic Web databases is presented here.

1. **SDB:** SDB is a subsystem of Jena that is design to support the scalable storage and query of RDF and OWL data using conventional SQL databases [Stocker et al., 2008]. SDB is designed specifically to support SPARQL.

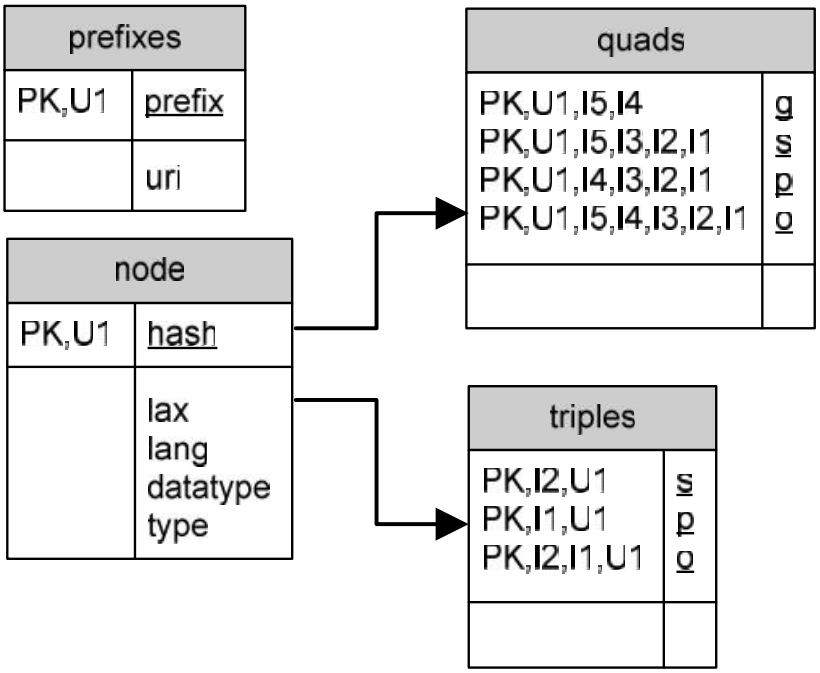


Figure 2.11: SDB Layout in MySQL

SDB follows the 'schema oblivious' approach as storage schema does not change even if the schema of the data to be stored changes [Stegmaier et al., 2009]. SDB always creates four tables in database with table name; Prefixes, Nodes, Triples, and Quads. The storage schema factors out the common prefixes of URI to reduce the storage space. It creates a separate table named as "Prefixes" to store the association of unique identifiers with each distinct prefix as shown in Table 2.1. SDB uses an id-based approach for triple storage that requires an additional table "Nodes" for storing one to one mapping between lexical values and corresponding identifiers as shown in Table 2.2. SDB supports two types of node tables depending upon the layout choice. In case of Index layout, a separate id is used beside hash values of the node. In hash layout, hash of the node is used as a node identifier. Figure 2.11 is presenting the hash layout. In hash layout, "Triple" table stores the hash value as an id for all nodes of the triples as shown in Table 2.3 and lexical values for these triples are stored in nodes table. "Quads" table store the ids of dataset quads for maintaining the provenance information as shown in Table 2.4. SDB creates one index on both of Prefixes and Nodes table, three indices for Triples table and six indices for Quads table. Detailed Description of each index is shown in Table 2.5.

Table 2.1: Prefixes table Design

Prefixes		
Column	**Type**	**Description**
Prefix	varchar (50)	Prefix for the corresponding asserted uri
Uri	varchar (500)	Asserted URI
Primary key: prefix		

Table 2.2: Nodes table Design

Nodes		
Column	**Type**	**Description**
hash	bigint(20)	CRC32 hash value for the asserted node
Lax	Longtext	Actual value of the asserted node
lang	varchar (10)	Language Identifier for the nodes if literals
datatype	varchar (200)	Data type for the nodes if literals
type	int(10) unsigned	Type of asserted node to differentiate blank nodes, literals and URIs. Type will be 1 for blank nodes, 2 for URIs and 3 for Literals.
Primary key: hash		

Table 2.3: Triples table Design

Triples		
Column	**Type**	**Description**
S	bigint(20)	Hash value for the subject of asserted statement
P	bigint(20)	Hash value for the predicate of asserted statement
O	bigint(20)	Hash value for the object of asserted statement
Primary key: S,P,O		

Table 2.4: Quads table Design

Quads		
Column	**Type**	**Description**
G	bigint(20)	Hash value for the graph of asserted statement
S	bigint(20)	Hash value for the subject of asserted statement
P	bigint(20)	Hash value for the predicate of asserted statement
O	bigint(20)	Hash value for the object of asserted statement
Primary key: G,S,P,O		

Table 2.5: Detail of Indices on SDB tables

Index on Prefixes Table			
Key name	**Column name**	**Non/unique**	**Index type**
Primary	prefix	Unique	Btree
Index on Nodes Table			
Key name	**Column name**	**Non/unique**	**Index type**
Primary	Hash	unique	Btree
Indexes on Triple Table			
Key name	**Column name**	**Non/unique**	**Index type**
Primary	s,p,o	Unique	Btree
ObjSubj	o,s	Non-unique	Btree
PredObj	p,o	Non-unique	Btree
Indexes on Quad Table			
Key name	**Column name**	**Non/unique**	**Index type**
Primary	g,s,p,o	Unique	Btree
SubjPredObj	s,p,o	Non-unique	Btree
PredObjSubj	p,o,s	Non-unique	Btree
ObjSubjPred	o,s,p	Non-unique	Btree
GraPredObj	g,p,o	Non-unique	Btree
GraObjSubj	g,o,s	Non-unique	Btree

2. **Sesame RDB:** SesameRDB store stores its data in a relational database. It supports 'schema-aware', 'schema-oblivious', and 'hybrid' as storage layouts for RDF data. The database's table layout can be tweaked using the "Max number of triple tables" parameter. Schema-oblivious approach creates a "monolithic layout" with a single table that stores all statements, by setting maximum number of tables' parameter equal to one. Schema- aware approach creates a "vertical layout" that stores statements in a per-predicate table, by setting maximum number of tables' parameter equal to zero or a negative value. Hybrid approach creates predicate tables as well as a single triple table that are collectively equal to desired max number of tables. The Schema-aware layout has better query evaluation performance on most data sets, but potentially leads to huge amounts of tables, depending on the number of unique predicates in dataset. If the number of tables becomes too large, the database's performance can start to decrease or it can even fail completely. Vertical layout in Sesame RDB is shown in Figure 2.12.

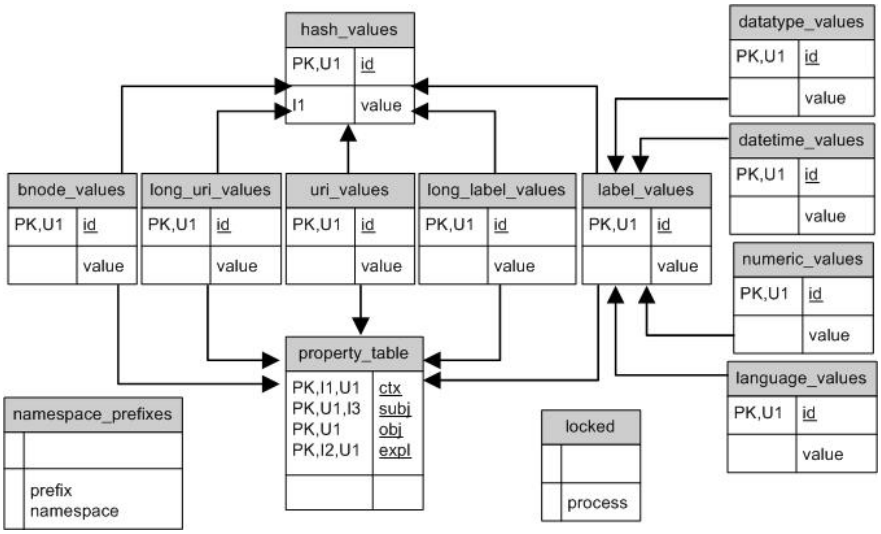


Figure 2.12: SesameRDB Layout in MySQL

As shown in Figure 2.12 SesameRDB always creates twelve fixed tables with table names; bnode_values, uri_values, label_values, long_label_values, long_uri _values, datatype_values, datetime_values, numeric_values, language_values, hash_values, namespace_prefixes, and locked. Along with these twelve tables it creates per-property tables in case of schema-aware approach or a single triples table in case of schema-oblivious approach. Each table along with fields and their data type is described in Table 2.6, Table 2.7, Table 2.8, Table 2.9 and Table 2.10.

SesameRDB an id-based approach for triple storage that requires some addi-

Table 2.6: Property table description

Property_table		
Column	**Type**	**Description**
Ctx	int(11)	Context of the data
Subj	int(11)	Id of the subject node for the triple with predicate of corresponding table
Obj	int(11)	Id of the object node for the triple with predicate of corresponding table
Expl	tinyint(1)	Value is zero if the triple is explicit and one if triple is implicit
Primary key: id		

Table 2.7: bnode_values/label_values/uri_values/long_uri_values/long_label_values /datetime_values/datatype_values/numeric_values/language_values tables description

bnode_values/label_values/uri_values/long_uri_values/long_label_value datetime_values/datatype_values/numeric_values/language_values		
Column	**Type**	**Description**
Id	int (11)	Ids of the {bnode/ label/ uri/ long-uri/ long-label/ datetime/ datatype/ numeric/ languages}, asserted into store as a part of statements
Value	$varchar(127)^1$ $longtext^2$ $bigint(20)^3$ $double^4$	Values of the $\{bnode^1/label^1/uri^1/long-uri^2/long-label^2/datetime^3/datatype^1/numeric^4/language^1\}$ asserted into store as a part of statements
Primary key: id		

Table 2.8: Hash_values table

bnode_values/label_values/uri_values/long_uri_values/long_label_value datetime_values/datatype_values/numeric_values/language_values		
Column	**Type**	**Description**
Id	int (11)	Ids of the {bnode/ label/ uri/ long-uri/ long-label/ datetime/ datatype/ numeric/ languages}, asserted into store as a part of statements
Value	$bigint(20)$	Hash of the $\{bnode/label/uri/long-uri/long-label/datetime/datatype/numeric/language\}$ asserted into store as a part of statements
Primary key: id		

Table 2.9: Namespace_prefixes table description

Namespace_prefixes		
Column	**Type**	**Description**
Prefix	varchar (127)	Prefix of the dataset
namespace	Text	Corresponding namespace of dataset

Table 2.10: Locked table description

Locked table description		
Column	**Type**	**Description**
process	varchar(128)	Process that has locked the database

tional tables i.e. uri_values, label_values, bnode_values, long_uri_values, and long_label_values, for storing one to one mapping between lexical values and corresponding identifiers. Whenever a triple is inserted into SesameRDB each node of the triple is assigned an id on the basis of its corresponding category i.e. uri, label, bnode, long_uri, and long_label; and the nodes are inserted into that corresponding table and triple is inserted into corresponding property table or in triples table in case of schema-aware or schema-oblivious approach respectively. Hash of each asserted node is computed and inserted into hash table along with its corresponding id. Hash values are used when looking up internal Ids from known terms. The uri_values, label_values, bnode_values, long_uri_values, and long_label_values tables store the URIs or literal text in value column, these cannot be indexed (in many databases), and so another index-able column is needed to lookup the internal ids for the text of URIs or literals. The hash value maps a globally unique 64 bit hash of the term to a local internal 32 bit id. Literal values of 'label_values' may have some rdf/xml properties such as datatype, datetime, numeric and language. These literals along with their corresponding ids and values are also inserted into the corresponding tables that are datatype_values, datetime_values, numeric_values and language_values, created as a part of schema design of the SesameRDB database. Namespace_prefixes table contains the namespace of the dataset and locked table contains the entry for each process that has currently locked the sesame database. SesameRDB creates four indices on each property table, two indices on hash table and one index on each remaining table (other than locked and namespace_prefixes). Detailed Description of each index is shown in Table 2.11.

Table 2.11: Detail of Indices on SesameRDB tables

Index on Property_Table			
Key name	**Column name**	**Non/unique**	**Index type**
Primary	ctx,subj,obj,expl	Unique	Btree
tableName_subj_idx	subj	Non-unique	Btree
tableName_ctx_idx	ctx	Non-unique	Btree
tableName_expl_idx	expl	Non-unique	Btree
Index on corresponding table			
Key name	**Column name**	**Non/unique**	**Index type**
Primary	id	unique	Btree
Index on hash_values table			
Key name	**Column name**	**Non/unique**	**Index type**
Primary	hash	Unique	Btree
HASH_VALUES_value_idx	value	Non-unique	Btree

2.2.3.3 In-Memory Stores

In-memory Semantic Web databases store and manage data in main memory they deploy different in memory data structures for efficient retrieval and storage of triples. A brief description of two in-memory Semantic Web databases is given in this section.

1. **Sesame - Memory (SesameM)** SesameM stores and manipulates Semantic Web data in in-memory. It uses a bipartite graph representation for triples in main memory. Vertices of this bipartite graph representation are divided into two parts. First part is a combination of statement nodes, representing the number of triples in the dataset. Second part is a combination of resource nodes, representing all resources, literals and blank nodes. Each statement node references four resource nodes, one reference for each resource role: subject, predicate, object or context. Each resource node also has, for each role it plays, a reference to a list of statement objects in which it plays that particular role. An overview of storage model is shown in Figure 2.13.

2. **Jena - Memory (JenaM)** JenaM is another prominent Semantic Web database that stores and manipulates Semantic Web data in main memory. It uses a HashBunchMap for triples storage and retrieval in main memory. At the heart it creates three indexes one for subjects to triples, second for predicates to triples and third for object to triples. These three indexes map RDF term (subjects, predicates, objects) to triple using a HashBunchMap. Jena uses its own hash maps which are more compact and faster than the standard Java ones although they provide fewer facilities, just what is needed for RDF indexing in Jena. In fact, the indexes map terms to "bunches" (triples with common indexing term) e.g. a subject index takes all the resources, that appears as a subject in triples, as hash-keys and stores all the triples as value for that key that contains that

Statements	Subject	Predicate	Object
1	Anila_Sahar	supervisedBy	Armin
2	Armin	supervises	Anila_Sahar
3	CS	hasMember	Armin
4	Anila_Sahar	memberOf	CS

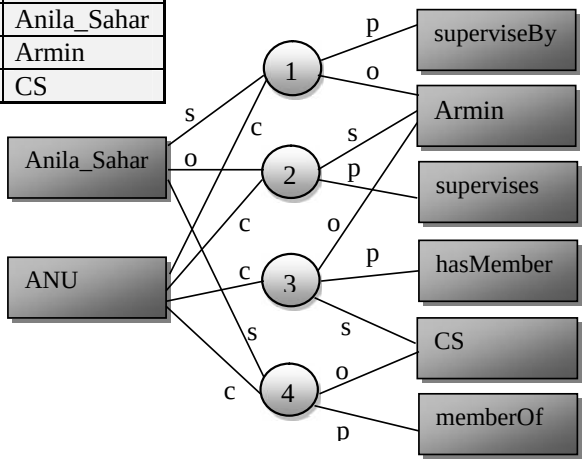


Figure 2.13: SesameM Storage Layout

resource as its subject as shown in Figure 2.14. This complete set of triples against any key is called a bunch and the bunch is stored as an array if small and a set (using the same code as Jena's hash maps) if larger. Similarly predicate and object indexes store triples with predicates and objects values as keys and store triples with having these keys as shown in Figure 2.15 and Figure 2.16.

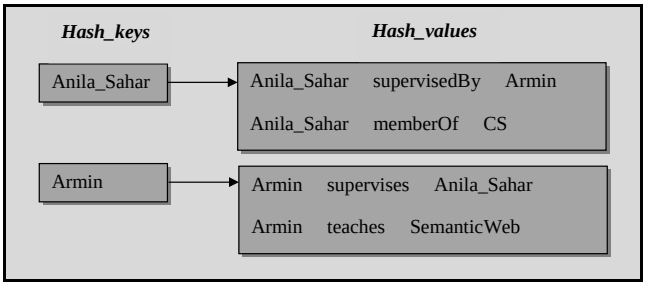


Figure 2.14: Jena Subject Index Structure

2.3 An overview of Semantic Web data retrieval process

Data retrieval on the Web is a complex process consisting of several steps. A common Semantic Web data retrieval framework is similar to a typical Web search process as shown in Figure 2.17. Here, boxes denote components of the data retrieval process, and lines indicate data flow among these processes. Most of the processes are concerned with the pre-processing phase (i.e. data acquisition, warehousing, indexing, and reasoning), while the remaining processes are concerned with the query-time

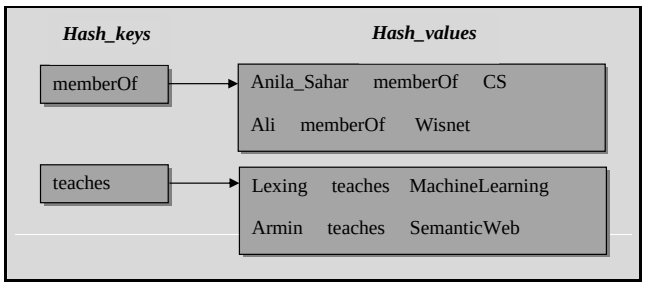


Figure 2.15: Jena Predicate Index Structure

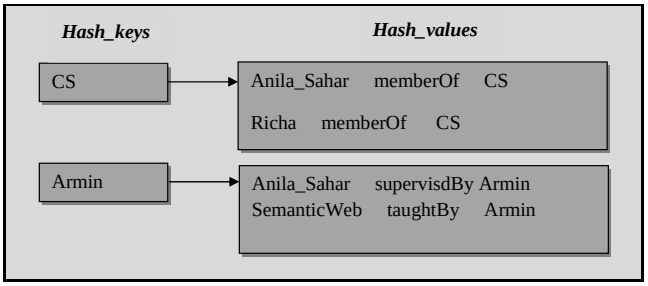


Figure 2.16: Jena Object Index Structure

processing phase, i.e., query evaluation. The ranking of search results may be part of the pre-processing or the query-time processing phase depending on the approach.

The first step, data acquisition (i.e. data crawling and parsing), is crucial for the retrieval approaches; because the quality of any retrieval system depends on the quality of the underlying dataset. Data acquisition necessitates Web crawlers, more specifically structured data crawlers for Semantic Web data crawling. The purpose of these crawlers is to gather a collection of linked data as quickly and efficiently as possible while providing at least the required features for respecting the limitations imposed by publishers (i.e. politeness and robustness). A large number of linked data crawlers has been proposed including [Batzios et al., 2008; Isele et al., 2010; Meusel et al., 2014]. These crawlers mostly gather data from the Web by traversing the linked graph. Moreover, some crawlers also clean the data syntactically. The output of the data acquisition process is materialised for further processing.

The graph-based nature of RDF(S) data necessitates special data structures for data storage. The Semantic Web community has proposed a variety of storage structures as discussed in Section 4.2.2.2. Some SWR approaches also infer implicit data (triples) from the crawled data before materialising it. To infer these logical consequences from the set of asserted facts or axioms, special-purpose reasoners are designed and reused by the community [Haarslev and Möller, 2003; Sirin et al., 2007]. Most of the research in this area (reasoning) is conducted separately, but not in the context or as a part of SWR approaches; therefore reasoning approaches are not covered in this work. However, there are existing benchmark studies that compare

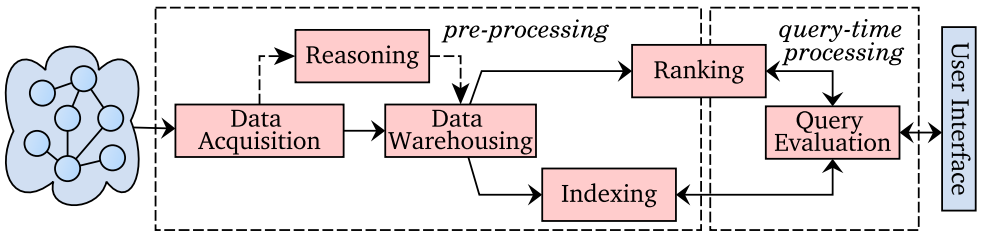


Figure 2.17: Outline of the general Semantic Web retrieval process

features and performance of the different available reasoners [Bock et al., 2008].

In large Semantic Web data collections, finding a match for a structured query or a keyword query requires lots of comparisons that are neither feasible nor necessary, because of infeasible query response times or a large number of non-matching triples in the data collection, respectively. Indexing techniques are required to mitigate this problem. A single word, a URI or a combination of URIs, commonly called the key, is used to decide where to find or insert data on disk. As with traditional Web information retrieval techniques, indexing has a trade-off between the computational complexity and the quality of the matching results. Having many small, but more specific keys in an index (more filtering) will lead to a smaller candidate result set and thus reduce the computational cost, but at the same time, it is more likely that some possible (partial) matches are being missed. On the other hand, a less specific key will result in a larger candidate result set but likely to lead to more exact matches. Various techniques for indexing linked data have been developed; an analysis on indexing, based on the structure and the content of the key is presented in Section 4.2.2.3.

In addition to providing information in response to a user query in real time through indexing, some retrieval approaches also provide a ranking of the results. The ranking tries to determine which result is the most appropriate for the query. A substantial amount of ranking models is designed or adapted for Semantic Web data ranking as discussed in Section 4.2.3. Some SWR techniques rank data in a data collection (i.e. corpus) off-line, independent of a user query, and materialise ranks along with indexing and others retrieve results for a query and apply ranking models to rank retrieved results only. Once the indexing and ranking are finished, the Semantic Web data becomes available for retrieval. Like Web search engines, SWR techniques allow users to explore linked data through keyword queries, but also through a structural query model where a user can pose additional more complex navigational queries. For this purpose, the user interface has to provide some means for a user to specify these complex queries. The users pose their queries through interfaces, and the queries are mostly evaluated in a bottom-up fashion. i.e., the first match of the content of a resource is found with the help of the index and then the other information is used for filtering and result selection through real-time queries.

2.4 Summary

This chapter presented the background material to understand the domain by outlining the techniques common in the Semantic Web. In the following chapter, we will review the literature in Semantic Web data retrieval techniques.

Chapter 3

Related Work

In this chapter, a review of related work in Semantic Web data retrieval is presented. We conduct a survey of 29 existing such techniques that are then characterise according to the taxonomy (as discussed is Chapter 4) in Table **??** to identify future research directions in Semantic Web data retrieval field.

3.1 Introduction

This thesis focuses primarily on ontology retrieval techniques. In this regard, we conduct a survey of existing Semantic Web data retrieval techniques and categorise them into three generations: (1) *Ontology Retrieval Techniques* that search ontology terms or ontologies, (2) *Linked data Retrieval Techniques* that perform search on linked or RDF data, and (3) *Graph Structured data Retrieval Techniques* that are designed for searching graph structure data in general and are suitable for ontologies and linked data search. Since ontologies, linked data, and graph data are based on *graph data models*, the survey also includes some prominent linked data and graph data retrieval along with ontology retrieval techniques. The approaches included in this survey are selected based on their relevancy, popularity and novelty. We refer to these techniques as Semantic Web data retrieval (SWR) techniques from this point onwards.

The centralised data retrieval techniques are the scope of this thesis; therefore, we exclude the decentralised search techniques, in this survey, where the primary focus is on designing distributed indexes, federated query processing, and the identification of duplicate data. In addition, specialized triple and quad stores e.g., Virtuoso[1], AllegroGraph[2], Jena[3], and Sesame[4] are not studied herein, because of the following two reasons: 1) A detailed discussion on the purpose, usage, type, and structure of such stores is provided in Section 2.2. 2) These stores create indexes and optimise queries for different SPARQL constructs and are not generally related to keyword-based retrieval and ranking, that is the focus of this thesis. Some performance studies on such stores have already been published in [Schmidt et al., 2009; Bizer and Schultz,

[1] http://virtuoso.openlinksw.com/
[2] http://franz.com/agraph/allegrograph/
[3] https://jena.apache.org/
[4] http://rdf4j.org/

2009; Butt and Khan, 2014].

We present surveyed SWR techniques under each category in chronological order to study how the techniques have developed over time. Each technique is given an identifier composed of either the technique name or the last name of the first author (in case a name is not assigned to a technique) and the last two digits of the year of publication, which is then used in Table **??** to identify an individual approach.

3.2 Ontology Retrieval Techniques

The techniques discussed in this section focus only on the retrieval techniques designed to find ontologies or ontological terms and are classified into three categories: (i) *Ontology libraries* - Web-based systems that contain a collection of ontologies and facilitate the task of searching and using these ontologies or different terms defined within, (ii) *Ontology search engines* - Web-based systems that automatically crawl the Web to index ontologies and provide search or reuse facility on these ontologies, and (iii) *Ontology Ranking Techniques* - techniques that are proposed in the literature and evaluated on an ontology collection but not available online.

3.2.1 Ontology Libraries

Ontology libraries are the Web-based systems that collect ontologies from different sources and facilitate the tasks of finding, exploring, and using these ontologies. A detailed analysis of ontology libraries is presented in [Noy and d'Aquin, 2012]. We select ontology libraries from a new generation of ontology libraries that has emerged in the last decade and give a brief description and analysis of each of the libraries in accordance to the taxonomy defined in Chapter 4.

OntoSelect04: [Buitelaar et al., 2004] is one of the biggest ontology library composed of more than 1500 *domain independent* ontologies. Ontologies are crawled from the Web or can be submitted by the users to make them part of the ontology collection. The crawler-based ontology collection made it a significant ontology collection (larger than CBRBench [Butt et al., 2014a]); however, the automatic ontology selection included many duplicates and outdated ontologies in the collection. It provides advanced search mechanisms (by keywords and by topic), with ranking mechanisms based on a set of measures on the ontologies (e.g., the number of imports, the number of languages included). It facilitates browsing ontology search results based on format and language of the ontology. OntoSelect focuses on providing ontologies and related services that facilitate annotation and natural-language processing in multiple languages, supporting searches that are restricted to a particular language.

OLS06: [Cote et al., 2006], the Ontology Lookup Service (OLS), developed at the European BioInformatics Institute, is a *domain-specific* library of biomedical ontologies, with the main goal of providing query and browsing access to a set of ontologies used for annotating biomedical data. OLS contains ontologies in the OBO format [Golbreich et al., 2007] federated by the OBO Foundry and added by the administrator into the OLS repository. It provides search features to find descriptions

of entities in these ontologies. The search is allowed within and across ontologies that result into *document-centric* result set without any order/ranking. Along with search, the navigation facility is provided to the user to explore ontologies. It also provides Web services access through SOAP services to the ontologies. The ontology collection of OLS is stable and numbered around 80. OLSVis[5] is an interactive visualisation of OLS and provides more features like faceted browsing.

The OBO Foundry07: [Smith et al., 2007] is a *domain-specific* ontology directory aims at collecting well-documented and well-defined biomedical ontologies that are designed to work with one another. It does not provide any search or browsing on ontologies; a SourceForge repository is used to maintain ontologies and ontology versions. The ontology collection, a stable dataset composed of nearly 80 ontologies, is established over time where users submit ontologies or ontologies are selected according to an editorial process defining which ontologies become part of that collection. The editor verifies the popularity, availability and uniqueness of an ontology for the ontology to be included in the library as an OBO Foundry candidate.

OntoSearch07: [Thomas et al., 2007] a *domain-independent* OWL ontology repository focuses on providing efficient querying mechanisms across hundreds of ontologies. It implements these querying mechanisms as an extension of the SPARQL language, reducing the set of ontologies into the DL-Lite formalism to enable the use of inferences in real-time, during the querying process. A stable ontology collection comprises of more than 200 ontologies, is collected after automatic syntax validation from the ontologies that are submitted by the administrator or user. The data is stored in a triple store and uses the underline indexing mechanism to retrieve SPARQL query results. Queries can be made through a simple *keyword-based* search form, or can be submitted as SPARQL queries, optionally containing fuzzy extensions that can specify the degree of confidence required for each term in the query. The keyword-based queries are expanded into fuzzy SPARQL queries, so all searches use the same internal process. The results are *document centric* and ordered and calculated by the text relevancy and position of keywords in the description of the resource. The ranking approach is similar to BM25F, originally designed for document ranking in information retrieval domain. The approach is evaluated on the LUBM benchmark [Guo et al., 2005] and proved to be more efficient and scalable than the baseline [Pan et al., 2006].

oeGov09 [Hodgson, 2009] is a *domain-specific* ontology directory, initiated by TopQuadrant [6], to create and collect ontologies used for e-Government. It relies on a blog system to get comments and reviews on ontologies. The scope is limited to ontologies only where access to a list or description of ontologies is made available without further search and browsing facilities. That makes oeGov a *document centric* ontology access. The oeGov dataset is stable, comprises of more than thirty ontologies at the moment, and is maintained by the administrators without additional indexes. The ontology collection is grouped into categories and ontologies are ordered alphabetically within each category.

[5] http://ols.wordvis.com/
[6] http://www.topquadrant.com/

BioPortal09: [Noy et al., 2009] is a *domain specific* ontology library to publish and explore biomedical ontologies; developed by the National Center for Biomedical Ontology. It has a growing ontology collection that comprises of almost 409 ontologies; collected through manual curation and automatic syntax validation along with the service for registered users to publish their ontologies in the BioPortal repository. Other salient features of BioPortal includes visualisation, search, browsing, recommendation and annotation of ontologies or resources (concepts and properties). It provides a nice visualisation of ontologies and concepts, naive *keyword search* and advanced search with added parameters across and within ontologies. The approach is empowered by *faceted-browsing* on ontologies based on five facets: *'Entry type'*, *'Duration'* - the lifetime of an ontology in the BioPortal repository, *'Category'* - subcategories within biomedical domain, *'Group'* - developers of ontology, *'Format'* - RDFS, OWL, OBO and UMLS , recommendation about relevant ontologies based on user-provided text, and annotations on concepts for a given text. The approach provides a web-based interface and REST service access to the ontologies and their metadata. The REST Web services is a Java implementation that is accessible in Java, Python, Ruby and few other languages. The search results are *document-centric* that are ordered based on the *page-view count* [Noy et al., 2013] as *focused* ranks.

The TONES repository10 [ton, 2010] is a purpose built *domain independent* OWL ontologies directory mainly built with the purpose of using these ontologies by tool developers for testing purposes. It has a stable ontology dataset, composed of roughly 230 ontologies, collected and maintained by an administrator. The purpose of the TONES repository (i.e. testing ontology editors or explorers) is achieved by carefully selecting the ontologies having the following characteristics: variation in size, the span of the range of expressive power, and exhibit different features. The TONES repository provides access to ontologies through a web-based interface and a Java RESTful Web services to access ontologies within any tool. The scope of the TONES repository is limited to browsing ontologies only, without further facilitating a search or browsing within the ontologies (i.e. concepts and properties). The interface provides access to the list of ontologies and faceted browsing based on features of Description Logic (DL) expressivity; that make it possible to browse ontologies according to parameters that determine their complexity (e.g., the DL expressivity and the number of logical axioms). The filtered results for faceted browsing are *document centric* without any specific ordering or ranking of the results.

NCBO10 [Jonquet et al., 2010] The National Center for Biomedical Ontology (NCBO) proposed a biomedical ontology recommender web service to suggest the most appropriate ontologies required to annotate a given biomedical text data. The recommender service recognises relevant concepts from an ontology repository to annotate the given text data, and then expands the first set of annotations using the UMLS Metathesaurus and the NCBO ontologies. The relevance of an ontology is computed based on the context and matching terms in that ontology; that mainly depends on the accuracy of the NCBO annotator.

LOV14: [Vandenbussche and Vatant, 2014] The Linked Open Vocabularies (LOV)[7], initiated in March 2011, is to the best of our knowledge, the most recent purpose-built ontology library available on the Web. LOV dataset contains *domain independent (generic)* RDFS or OWL ontologies used or usable by datasets in the Linked Data Cloud. LOV dataset ontologies are those published and freely available on the Web and small enough to be easily integrated and reused, in part or as a whole, by other vocabularies. LOV search scope is limited to *ontological terms or ontologies*, and search results are *document-centric*. The approach facilitates users to find relevant concepts and properties based on *keyword* queries, limited *structured* queries (SPARQL DESCRIBE and SELECT queries only) on meta-data, and a näive faceted browsing based on three fixed facets: `'Type'` of ontological term, `'Tag'` mostly used for the result terms and `'vocabularies'` the search results belongs to. The 'LOV Aggregator' aggregates the *admin selected* ontologies in a single endpoint. A *lucene index* is built that facilitates the efficient retrieval. The query is evaluated using *index* for keyword queries, and SPARQL endpoints for *structured* queries. It uses a *global* ranking algorithm based on the term *popularity* in Linked Open Data (LOD) and in the LOV ecosystem; the ranking model is rooted in an *information retrieval* ranking model for document retrieval.

BiOSS14 [Martínez-Romero et al., 2014], a system for the selection of biomedical ontologies, outputs single or combined matching ontologies for a user specified keyword. BiOSS proposes three metrics to evaluate and rank matching ontologies: 'domain coverage' - the extent the query terms are described by each candidate ontology, 'semantic richness'- the richness of concepts that are involved in the domain coverage, and 'popularity'- a count of references to the ontology from each Web 2.0 resource. The output matching ontologies are ordered according to the aggregated scores combined from these metrics. Although the evaluation results show it to be the complete biomedical ontology library, the methods to compute its evaluation metrics are not applicable for general ontologies (e.g. the use of the UMLS metathesaurus, and PubMed and BioPortal to compute semantic richness and popularity, respectively).

Cupboard [d'Aquin and Lewen, 2009] ontology library is built around the idea of spaces, enabling each user to create her own ontology space, containing and relating the ontologies she has selected. Cupboard provides a number of functionalities in the context of these ontology spaces, supporting ontology developers in publishing and sharing their ontologies, ontology users in finding and reusing ontologies, and application developers in using ontologies in their applications. In particular, Cupboard provides users with the ability to search in dedicated ontology spaces and across the entire library, to rate and comment ontologies, and to relate ontologies in an ontology space through alignments. In addition, a plugin for the NeOn Toolkit ontology editor allows ontology developers to reuse elements of shared ontologies directly within the ontology engineering environment.

[7] http://lov.okfn.org

3.2.2 Ontology Search Engines

Ontology search engines are the web-based systems that crawl ontologies from the Web and enable search on the crawled ontologies. The prominent ontology search engines are surveyed below:

OntoKhoj03: [Patel et al., 2003] is a *domain-independent* crawler-based Semantic Web portal that crawls ontologies over the Web, indexes and ranks these ontologies in a local repository, classifies each of the stored ontology under a particular category and then makes it visible to the user to retrieve and reuse. It's distributed crawler performs ontology crawling on heterogeneous Web sources and aggregate RDF chunks that belong to the same URI. The portal uses a machine learning based approach [bow, 1998] to classify each ontology to the most appropriate domain for which the ontology has been engineered originally. A *PageRank* [Page et al., 1998] based ranking model is used to rank a large number of ontologies within each category. This ranking is performed globally for all ontologies within each category irrespective of the query. OntoKhoj provides *document centric* search through a *Web based GUI* and *API* as a user interface. The GUI extends keyword queries by allowing users to choose the correct sense (made available through wordnet incorporation in the interface). The API facilitates agents to access the ontology directory and traverse and retrieve desired information, and through logical query interfaces (e.g. RDQL and FLogic).

Swoogle04: [Ding et al., 2004] Swoogle is a crawler-based indexing and retrieval system for the Semantic Web, i.e., for Web documents in RDF or OWL. Swoogle is mainly concerned with a more traditional document-search over ontologies. It provides a web-based interface to search and browse Semantic Web data. It automatically discovers Semantic Web documents, extracts meta-data, and computes relations between documents. It further discovers URIref, either of document or term, from the crawled documents and hashes them to a token before indexing in an inverted keyword index on URIs. Swoogle facilitates *keyword search* over RDF documents using an inverted keyword index and a relational database. A user can search an ontology and/or terms, referred to as a Semantic Web Document (SWD) and a Semantic Web Term (SWT) in their work, and the search is *document centric*. It calculates metrics that allow ontology designers to check the *popularity* of certain properties and classes and orders results according to their popularity. The ranking model to compute popularity is rooted in an *Information Retrieval* method to rank documents (i.e. PageRank [Page et al., 1998]). The ranks are computed *globally* independent of the query terms.

WATSON11: [d'Aquin and Motta, 2011] Watson is a *domain-independent* Semantic Web search engine that collects available semantic content on the Web (both ontologies and entities), analyses it to extract useful metadata, indexes the data and then implements querying facilities to access the data. It uses Heritrix[8], an Internet archive crawler, to crawl the existing Semantic Web documents and then indexes based on Apache Lucene indexing. Query evaluation is done by these indexes. It provides a Web-based *GUI* and *APIs* (both RESTful and SOAP web services) that facilitate the

[8] https://webarchive.jira.com/wiki/display/Heritrix

search for and within ontologies and Semantic Web documents, and explore metadata about them. The WATSON API is implemented in Java using Jena, a Semantic Web triple store. The users are allowed to make *keyword-search* and *SPARQL* queries through both interfaces and are returned results in *document centric* and *relation centric* fashion that are further grouped based on the ontologies they belong to. The ranking offered by Watson relies on a combination of some quality measures (i.e., structural measures, topic relevance, etc.), that are stored along with the ontologies. These measures are made available to the user or application to design their ranking scheme according to the application requirement.

3.2.3 Ontology Ranking Techniques

In this section, we present some ontology retrieval techniques that have been proposed in the literature with the focus on ontology ranking.

AKTiveRank06: [Alani et al., 2006] proposed a *document-centric* ontology retrieval technique with prime focus on appropriate ordering of the search results. It is a *domain-independent* approach that facilitates in finding the most suitable ontologies, without facilitating search within ontologies. The approach uses keyword/s to find the relevant set of ontologies from the underlying Semantic Web search engine (i.e. Swoogle [Ding et al., 2004]) and then applies four ranking models [Freeman, 1977; Rada et al., 1989; Spanoudakis and Constantopoulos, 1993], originally proposed for *graph retrieval* or *document retrieval*, on the task of ranking ontologies. The ranking factors considered to order the search results are *coverage* and *semantic similarity*. The effectiveness of the approach is evaluated against Swoogle in a user study and found effective as compared to Swoogle.

SAB06: [Sabou et al., 2006] is another *document-centric* and *domain-independent* ontology selection technique that considers a query text, retrieves triples out of the query text, expands query terms by considering their synonyms and hyponyms, identifies the matching ontologies and ranks them for the given query text. The ranking model relies on the generality deviation of a matching ontology from the query triples. The ranking quality heavily depends on the query expansion and triple extraction processes.

WebCORE07: [Cantador et al., 2007] recommends the most appropriate ontologies for a given domain. The tool allows a user to refine and enlarge query terms using WordNet, and then automatically recommends ontologies based on a query terms' frequency in the matched ontology and the knowledge base. WebCORE modifies the vector space model to compute the similarity between the query vector and ontology vector. Moreover, it considers the manual user evaluations for ontology's *correctness*, *readability*, *flexibility*, *level of formality*, and *type of model* in order to incorporate a human, collaborative assessment of ontologies. The ranking of an ontology for a user is measured as the average of its five evaluations.

OntoQA07: [Tartir and Arpinar, 2007] is a *document-centric* ontology retrieval technique that evaluates ontologies related to a certain set of terms and rank them according to a set of metrics. The evaluation of an ontology is made on two dimen-

sions: Schema and Instances. The schema metrics include the 'relationship diversity'- ratio of non-inheritance relationships to total number relationships, and 'schema deepness'- average number of sub classes per class. The instance metrics measures the effectiveness of a schema in terms of the placement and distribution of instance data. The final relevance score of a candidate ontology is the weighted average of the schema and instance metrics; weights are set based on empirical testing. The evaluation of the proposed approach is presented for three ontologies only.

3.3 Linked/RDF data Retrieval Techniques

The second generation of Semantic Web data retrieval techniques mainly focuses on the approaches to find linked data or RDF data. The techniques under this generation belongs to two categories: 1) *Efficiency oriented techniques* - these techniques focus on design and implementation of efficient index structures for fast data retrieval, and 2) *Effectiveness oriented techniques* - the focus of such techniques is more on the design and implementation of ranking algorithms to improve the quality of the retrieved results.

3.3.1 Efficiency oriented techniques

Some linked data retrieval techniques target to improve the *efficiency* of the retrieval process. These techniques propose special purpose indexes and query optimisation to improve the query response time. Below we survey some linked data retrieval techniques that automatically crawl the RDF data from the Web, index that data and provide data retrieval facility from the index through queries.

Sindice08: [Oren et al., 2008] is a lookup index over resources crawled from the Semantic Web. This index allows software agents (i.e. applications) and users through a user interface to search documents that contain information about a given resource. The Sindice crawler crawls the Semantic Web and indexes all identifiers (i.e. URIs and IFPs - Inverse functional properties) and literals (i.e. keywords) in a document. The three indexes store resource URIs, resource IFPs and literals, and their occurrences in RDF documents. The URI index contains an entry for each resource URI that lists the document URLs where this resource occurs. The IFP index allows searching such resources that have different URIs but identify the same real-world concept. For this, the IFP index entry records a uniquely identifying property-value pair as an index key that points to a list of document URLs where this pair occurs. The literal index is built after extracting tokens from the literals in the documents. An index key in the literal index is an entry for an extracted token that points to a list of document URLs. The indexes are optimised for disk space and lookup times. An inverted index structure is adapted to provide access from resource to mentioning sources. These indexes have been implemented both as an on-disk persistent hashtable, mapping from resource URIs to mentioning documents and in the Solr information retrieval engine.

Sindice ranks matched documents according to their relevance for the search resource. It implements a ranking criterion for results, named A- box ranking technique, that works on focused sub-graphs (result sets) only. The results are ranked with the use of very little meta-data about the sources to make this process relatively faster to compute. The ranking score is computed based on the rare (i.e., less common) share terms between the requested terms (query string) and filtered results (URIs, IFPs, keywords) rather than common terms. This score is collected from a built-in scoring function of the inverted index of Sindice.

Sig.ma10: [Tummarello et al., 2010] is a Semantic Web data search service powered by Sindice [Oren et al., 2008]. Sig.ma consolidates heterogeneous data gathered on the Web into a single entity profile using Semantic Web data consolidation techniques. The user can find these entity profiles as live views on the web, she can enhance the information with additional data sources and can reuse it in other Semantic Web applications. A user can search for a key phrase or a single resource identifier. This resource is searched on the Sindice index, it then returns documents which mention a certain identifier. Sig.ma then consolidates the data from different documents. Sig.ma depends mainly on the indexing and ranking of its underlying search engine (i.e., Sindice). It implements few ranking improvements during the alignment of related sub-graphs, like ranking the resources high that are defined and used in the same graph.

SWSE11: [Hogan et al., 2011] is a Semantic Web search engine that crawls and indexes Semantic Web data. SWSE searches for an entity based on a keyword search query over its index and consolidates the information about that entity from multiple sources and returns ranked result to the user. For ranking, SWSE uses a PLD level naming authority variation of centralised ranking technique [Hogan et al., 2006] in a distributed environment. The final ranks are calculated by combining these ranking scores with the relevancy score of an entity. In a distributed framework based on a share-nothing architecture, each slave machine scans its data and extracts naming PLD level links for authority graph. A master machine aggregates a PLD graph and executes the PageRank algorithm and sends PLD scores to all slave machines. The slave machines compute the identifier rank based on the PLD scores. The master machine then gathers these identifier ranks from all slave machines and computes the global ranking for the identifiers.

3.3.2 Effectiveness oriented techniques

Some linked data retrieval approaches focus on the effectiveness of the search results. These approaches propose or adopt ranking models to find the best-matched results for a user query.

SemRank05: [Anyanwu et al., 2005] presented a *property-centric* modulative approach for ranking semantic associations (i.e. relationships among resources) . The approach mainly addresses the issue of determining the relative importance of different relationships found between two resources from an RDF store or a graph database [Anyanwu and Sheth, 2003]. The retrieved results can be ranked based on

two contexts: *'Conventional'* and *'Discovery'*. A user can select a range on a scale from 0 to 1. 0 is for the conventional or implicit relationships search (predictable from the schema) between two resources. Whereas 1 if for the discovery search that involves ranking unpredictable or explicit results higher in result set. Unpredictable associations are those that deviate from the schema associations; they arise because of the multiple typing of resources by class inheritance allowed by RDF and OWL-Lite (such as a property can have multiple domains and ranges). The SemRank relevance model uses a blend of *semantic technique* (i.e. refraction), *information theoretic technique* (i.e. information gain) and S-Match (i.e. a heuristics approach) to determine the rank of semantic associations [Giunchiglia et al., 2004]. A prototype of the proposed approach is developed for an empirical evaluation on a synthetic dataset; however, it is not accessible through a web interface or an API.

ReConRank06: [Hogan et al., 2006] presents an A-Box ranking technique that adopted PageRank for Semantic Web data ranking while taking into account the context of data. This approach is proposed as part of the Semantic Web Search Engine (SWSE) [Hogan et al., 2011]. The proposed ranking model works on topical sub-graphs (focused graphs) composed of matched results, and not on the whole RDF graph. The idea was to avoid the computational efforts for rank computation for entire RDF graph and rank update cost when the index is updated as a result of newly added triples. The ReConRank approach computes a ResourceRank for each resource in a topical sub-graph, using PageRank with few modifications, to prioritise the relevant results according to their popularity. Moreover, ContextRank is computed by applying the ranking algorithm (i.e. PageRank) to context graph in topical sub-graph. ContextRank identifies the trustworthy sources of data. ContextRank further powers the ResourceRank of each resource by computing the amount of data of this resource provided by that context. A combination of ResourceRank and ContextRank i.e. ReConRank ultimately prioritised the search results according to their importance and trustfulness. The ReConRank is a component of overall data retrieval framework; that is placed between the URI component and the index structure. A focused sub-graph that is composed of matching triples to a query string is selected as a result of the query processing over the indexed data. The size of this graph can be tweaked by setting the number of hops (classes or nodes) that are allowed from a directly matched subject in the graph. Initial weights are estimated on the basis of the ratio of all incoming links of a class to total links of topical sub-graph. Between every fifth iteration, a quadratic extrapolation is used to speed up convergence. The idea of assigning weights to the link, according to the user preference, was proposed, but not implemented. For the ContextRank computation, a notion of implicit links between resources and contexts, context and resources, and context and context is introduced, but not fully implemented in work.

RSS08: [Ning et al., 2008] proposed a framework to rank search results on the Semantic Web. In this work, the authors exploites the heterogeneity of relationships to determine the importance/relevance of resources to a user query. The proposed search algorithm first selects the nodes (entities) that contain one or more query keywords along with the properties, and then the search results are expanded by

including more entities, in each result, that are semantically related to the query terms. A ranking model is proposed that computes ranks in two steps: (1) A-Box ranking of entities (referred to as global ranks) is computed considering the T-Box at the global graph (whole dataset) using PageRank. The links (properties) between two nodes (instances) are assigned weights depending upon the weight of the property in the ontology, whereas the weights assignment to properties in an ontology is a manual and continuous trial. The global rank for each node (entity) is computed with a little modification in the PageRank algorithm. The initial eigenvector (matrix) for PageRank is computed on the basis of these weights rather than using a common initialization value, and the final global rank is computed till convergence through PageRank algorithm. (2) A final rank is computed on a topical graph (a matched solution) by using the spreading activation method. Initial activation values are set by combining the global rank (computed in step 1) of the resource with their relevancy score with the search term.

TripleRank09: [Franz et al., 2009] is a Semantic Web data ranking method proposed as a part of the Semantic Web browser LENA. This is an A-Box ranking technique for Browser-based search semantics (relations). It ranks the properties and corresponding values for each resource that appear as a subject in the data. This ranking is more about the faceted ranking in RDF graphs. The ranking is performed globally. The TripleRank takes into consideration a two-dimensional graph representation and represents Semantic Web data by a three-dimensional tensor where each of its slices represents an adjacency metrics for one RDF property of dataset. It further applies PARAFAC decomposition of tensors that results into relevancy and connectivity score for links. This framework is appropriate for faceted browsing of semantic data rather than keyword or arbitrary relations ranking. For each resource, TripleRank shows ranked properties as facets and ranked object values for those facets.

Harth09: [Harth et al., 2009] The work presented in this is an improvement of authors' previous work [Oren et al., 2008]. The algorithm ranks the structured data which have been integrated from multiple data sources about named entities, without prior knowledge of the schema. A named entity on the Semantic Web can have multiple predicates and multiple object values for each predicate scattered in various sources. The focus is on the problem of prioritising predicates and then objects for multi-valued predicates. This problem necessitates both the T-Box (predicates) and A-Box ranking (objects). This technique introduces the notion of 'naming authority'. The naming authority of a global identifier (URIs excluding blank nodes) is the data source which defines this identifier. Use of a URI owned by data source 1 in data source 2 is a positive vote from the data source 2 to 1. The algorithm first derives a naming authority graph from the input data set. This graph encodes the links between data sources based on the implicit connection made due to the use of identifiers in a data source other than its naming authority. Then it computes the ranking score by implementing a PageRank algorithm over the naming authority graph. Now, when all the data sources have their rank scores, an individual identifier's rank is calculated by adding ranks of all the data source where this identifier

occurs. The rank of blank nodes and literals are defined as the source rank in which they occur. Two variations for naming authority are proposed; the complete URI of a document (global URI) or Pay-level domains (PLDs). PLD groups many URIs into a single naming authority, thus it reduces the size of the naming authority graph. The evaluation is conducted for four different variations of this algorithm against PageRank over the object graph. The versions includes 1) All links contribute authority & URI level naming authority (AU) 2) External links contribute authority & URI level naming authority (EU) 3) All links contribute authority & PDL level naming authority (AP) 4) External links contribute authority & PDL level naming authority. A user study conducted for evaluation shows that all these four versions perform better than PageRank in terms of ranking quality.

DING10: [Delbru et al., 2010] is a hybrid algorithm that combines both weighted and hierarchical link analysis models to rank the Semantic Web data. Like [Harth et al., 2009] it uses links among datasets (data sources) to compute data set ranks and exploits dataset ranks to calculate the entity ranking within each dataset. Unlike [Harth et al., 2009] it assigns weights to the links of dataset graph (naming authority graph in [Harth et al., 2009]) and entity ranking is not merely the frequency model rather it further considers link analysis within each entity graph i.e. the dataset. First, it generates the dataset graph from Semantic Web data crawled and indexed for Sindice [Oren et al., 2008]. Then a weight is assigned to each link in the dataset graph since the label and the number of links among different data sources are not the same. A link weighing function LF_IDF (Link Frequency - Inverse Dataset Frequency) is derived from TF_IDF, which measures the relevance of a label given its frequency in a data collection. LF_IDF assigns a higher degree of importance to a link with a high frequency and a low dataset frequency in the dataset collection. Next, DatasetRank is computed by reformulating the original PageRank algorithm (rank of a back-link node is multiplied with back-link weight). The algorithm then computes the rank for each entity within the dataset called EntityRank either by 'Weighted Entity Rank' or 'Weighted Link Count'. The Weighted Entity Rank algorithm uses the PageRank algorithm along with the LF_IDF scheme to compute the EntityRank for each entity within the dataset. Weighted Link Count is a single iteration rank computation algorithm for EntityRanking, where the rank of each node in a dataset is the sum of weights of all in-links (backlinks) of that node, while the rank of in-links nodes is ignored. Finally, DatasetRank and EntityRank are multiplied to get the global score for each node in Semantic Web data.

Dali12: [Dali et al., 2012] presents an RDF data ranking algorithm rooted in learning to rank technique. The author adapted a pairwise learning to rank approach, RankSVM. The approach proposes a global ranking algorithm where the rank of the resources are computed globally, and the query string does not effect on the ranking; therefore, it is referred to as query-independent search. The approach first introduced Learning to rank and RankSVM for RDF resources, it then identifies the features and analyses their impact on the ranking performance. Moreover, the authors also established a ground truth for analysing the performance of this approach. The features identified in this approach are grouped into dataset specific (extracted

from RDF graph) and dataset independent features. Features extracted from the RDF graph include: the number of subject (i.e., the number of times a node appears as subject), the number of object (the number of times a node appears as an object), the number of types of outgoing predicates (the count of unique outlinks), the number of types of incoming predicates (the count of unique inlinks), the average frequency of outgoing predicate, the average frequency of incoming predicates, the number of literals attached with each node, PageRank score of a node, Hubs and authority score for a node. Dataset independent features include search engines based statistics: the number of times a node label appears on the Internet in search results, and the number of times it occurs in google n-gram database.

3.4 Graph Structured data Retrieval Techniques

This section describes the approaches designed to retrieve subgraphs, for a keyword or a structured query, from graphs or graph type databases.

Yan05: [Yan et al., 2005] investigates the issue of substructure similarity search using indexed features in graph databases. The proposed method finds graph(s) that contain the query graph or a query subgraph (i.e., an approximate match) from a graph database. Since the pair-wise substructure similarity is a computationally expensive, a filter based measure is used to screen the database before performing expensive pairwise structure based measures. Therefore this work is a combination of feature-based and structure based measures to solve the problem of substructure similarity. The proposed approach identifies small structures as graph features in the graph database, and a feature graph matrix is built for these features in the database. When a graph query is executed on graph databases, the graph query features are extracted from query graph, and the upper bound of feature misses is calculated. By a maximum number of query features that can be missed, initial graph screening is done to reduce the search space. The feature-graph matrix is used to calculate the difference in the number of embedding of common features between each graph G in the database and query graph Q. If the difference is greater than the maximum allowed misses (i.e. misses more than allowed numbers of embedding) graph G is discarded. The remaining graphs constitute a candidate answer set. The candidate answer set is then checked for pair-wise substructure similarity to find the query results. The approach is evaluated on real data in an AIDS antiviral screen dataset and synthetically generated dataset that has labels on edges.

BLINK07: [He et al., 2007] proposes a bi-level indexing and query processing scheme for top-k keyword search on graphs. It follows a search strategy, regarding search within a graph or different blocks of the graph, available in literature and additionally exploits a bi-level index for pruning and accelerating the search. To reduce the index space, BLINKS partitions a data graph into blocks and use the bi-level index stores information. Block Indices store information about different blocks and Intra block indices stores summary information at the block level to initiate and guide search among blocks and more detailed information for each block to accel-

erate search within blocks. Before creating indexes, first the graph is partitioned into blocks using a *node separator partitioning* approach. Two algorithms are proposed that first partition a graph using edge-separators and then convert edge-separators into node-separators. An **Intra-Block Index (IB)** indexes information inside a block. For each block 'b', the IB-index consists of the following data structures: Intra-block keyword-node lists, Intra-block node-keyword map, Intra-block portal-node lists and Intra-block node-portal distance map. A **Block Index** indexes information of different blocks: keyword-block lists and portal-block lists. When a search is made for keywords w_i, the `keyword-block list` is used to find blocks containing w_i. A cursor is used to scan each `intra-block keyword-node list` for w_i, these cursors are all put in queue Q_i. When an in-portal node u is reached of the current block, backwards expansion is continued in all blocks that have u as their out-portal. Such blocks can be identified by the `portal-block list`. For each such block b, expansion is continued from u using a new cursor, this time, to go over the `Intra block portal-node list` in block b for out-portal u. Each time the cursor is initialised with a starting distance equal to the shortest distance from u to w_i. The cursor will automatically add this starting distance to the distances that it returns. Thus, the distances returned by the cursor will be the correct node-to-keyword distance instead of the node-to-portal distances in the portal node list. The `intra-block node-portal distance map` is consulted for the shortest distance from u to any out-portal in its block. If this distance turns out to be longer than the intra-block distance from u to w_i, it is concluded that the shortest path between u to w_i lies within the block. The approach is evaluated on the DBLP and the IMDB datasets for ten keyword Queries ranging from 2 to 4 keyword.

Lindex2011: [Yuan and Mitra, 2013] proposed a lattice-based index, applicable on all graph features, to improve the filtering power and query processing time for subgraph searches. Lindex is a graph index which indexes subgraphs contained in database graphs. A node in Lindex represents key-value pairs where the key is a subgraph in a database, and the value is a list of database graphs containing the key. A directed edge between two indexed nodes indicates that the key in the parent node is a subgraph of the key in the child node; the edge holds the transitive property. The index is constructed on the basis of two heuristics to answer the subgraph queries that enables less subgraph isomorphism tests to improve the efficiency of subgraph-querying. It utilises the fact that database graphs that contain a supergraph of a query are guaranteed to be the answer set for the query; these graphs do not need to be checked for subgraph isomorphism. To further reduce the candidate set, Lindex+ is proposed to support indexing frequent graph queries on disk [Yuan and Mitra, 2011]. Lindex+ contains two parts: an in-memory Lindex and a set of on-disk Lindexes. The in-memory index is a first level Lindex. These two parts of Lindex+ is used to partition the value set such that subgraph isomorphism tests need to be performed on database graphs appearing in one partition resulting in further reduction in the candidate set. The benefits of Lindex and its disk-resident variation Lindex+ are evaluated theoretically and empirically. The AIDS dataset comprises of 40,000 graphs used to evaluate the effectiveness of (1) filtering false graphs, (2) index

lookups, (3) index construction and maintenance, and (4) indexing large feature set.

Wu13: [Wu et al., 2013] presents an *ontology based subgraph querying technique* that measures the similarity of the nodes by exploiting the ontology graphs. They further introduce a *metric* to identify topK matches of a query graph *Q* in graph database *G*. An *ontology-index* is built to propose a *filtering-and-verification* framework for computing top-K matches. The framework is composed of three components. *Index construction:* The framework first constructs an ontology index for a data graph G, as a set of "concept graphs". Each concept graph is an abstraction of G by merging the nodes with similar labels in the ontology graph. The index is precomputed once and is dynamically maintained upon changes to G. *Filtering:* Upon receiving a query Q, the framework extracts a small subgraph as a compact representation of all the matches that are similar to Q, by visiting the concept graphs iteratively. If such a subgraph is empty, the framework determines that Q has no match in G. Otherwise, the matches can be extracted from the subgraph directly without accessing G. *Verification:* The framework then performs isomorphism checking between the query and the extracted subgraph to extract the (top K) matches for Q. The approach is tested on both real-life graphs and synthetic data. An extensive evaluation is done to evaluate the effectiveness, flexibility, efficiency, scalability and effectiveness of ontology index.

NeMa13: [Khan et al., 2013] is a neighbourhood-based subgraph matching technique for querying real-life networks. It converts the underlying graph isomorphic problem into an equivalent inference problem in graphical models and therefore applies an inference algorithm to heuristically identify the optimal matches. To measure the quality of the match, a subgraph matching cost metric is introduced. The metric aggregates the costs of matching individual nodes and unifies both structure and node label similarities. However, a match may not necessarily be isomorphic to the query graph in terms of the label and topological equality. Therefore, in contrast to strict subgraph isomorphism, the proposed cost metric aggregates the costs of matching individual nodes, which in turn depends on the cost of matching node labels and their neighbourhoods within certain hops. On the basis of the metric, the minimum-cost subgraph matching problem is defined. For a query graph on a graph database, the goal is to find the top-k matches of the query graph with minimum costs in the target graph. The effectiveness, efficiency, and scalability of the approach is tested on three datasets: 1) IMDB Network, 2) YAGO Entity Relationship Graph, and 3) DBpedia Knowledge Base. The effectiveness is measured through Precision, Recall and F1-Measures.

SLQ14: [Yang et al., 2014] The main idea of this work is to provide a flexible graph querying interface for non-professionals. The proposed SLQ (**S**chema**l**ess and **S**tructure**l**ess **Q**uering), a graph querying interface, is built on a set of transformation functions (i.e., synonyms, antonyms, and string functions) that automatically map keywords and linkages (nodes and edges) from a query to their potential matches in a graph and returns top-ranked matches. There might exist multiple matches for Q in graph G using different transformations. To identify the top k matches (results) for a given query, the matching quality of the corresponding match to the

query is determined by aggregating the quality of all matched nodes and edges in a given match. Intuitively, a direct match should always get a high matching quality score, or the matching quality should be determined by the importance of the transformation. However, rather than guessing or assigning weights (importance score) to each transformation manually, a learning approach is introduced to figure out the weights of the transformation. SLQ learns weights of transformations in an `off-line model learning` and the topK results are selected on the basis of these weights during `Online query processing`. Three indexes are created to find transformed matches of query quickly: (1) String Index (StrIdx) - a list of (key, value) pairs, where key is a label and value is a list of nodes, such that each node in value list has the label (with or without transformation). Each string transformation is applied on each label of all the nodes in G. The transformed labels are set as keys of StrIdx and node is added to the value list of that label, (2) Semantic Index (OntIdx) - identify matches based on semantic information (ontology and synonyms) for each node and add it to the OntIdx. (3) Numeric Index (NumIdx) - a B+ tree constructed over numeric values for the labels with numeric values. e.g. every numeric value node less than 35 will be the value for node $\leq$ 35. The approach is evaluated on DBPedia, YAGO2 and the Freebase datasets for DBPSB Benchmark 25 queries (simple with tree hierarchy) and self-generated graph query templates (cyclic) from datasets.

3.5 Summary

In this chapter, we present a survey of historical and current state-of-the-art techniques for Semantic Web data Retrieval. We will next characterise the techniques presented in this chapter according to the taxonomy we propose for Semantic Web data retrieval, identify the important aspects missing in the existing techniques, and analyse research directions in detail in the next chapter.

Chapter 4

A Taxonomy of Semantic Web data Retrieval Techniques

In this chapter, we address the first research question, i.e., "What are the limitations of existing approaches for ontologies retrieval?". In this regard, we present a taxonomy of Semantic Web data retrieval (SWR) techniques to characterise such techniques along sixteen dimensions. We discern existing techniques (surveyed in the previous chapter) along this taxonomy and summarise them in Table 4.1. We then highlight shortcomings of current techniques based on this characterisation and discuss avenues for research in Section 4.3.

4.1 Introduction

The domain of the Web retrieval techniques is covered by a number of reviews including [Broder, 2002; Sarawagi, 2008; Baumgartner et al., 2009]. [Broder, 2002] defines a taxonomy of web searches, which classify web queries based on the 'need behind the query'. [Laender et al., 2002] introduces a set of criteria and a quantitative analysis of various Web data extraction systems along with a taxonomy to categorise such systems. [Chang et al., 2006] presents a tri-dimensional categorization of information extraction systems, based on techniques used, task difficulties and degree of automation. To the best of our knowledge, the survey from [Ferrara et al., 2014] is the most recently updated review on Web information extraction. The survey provides an overview of the Web data extraction applications and techniques and categorised these techniques into *'Enterprise'* and *''Social Web'* applications. The authors also discuss the potential of Web data extraction techniques in other domains. All reviews mentioned above discuss Web retrieval techniques and do not analyse Semantic Web data (i.e. Graph-structured data) retrieval techniques.

In recent years, few surveys on Semantic Web data retrieval have been published. [Abadi et al., 2007] presents a performance analysis of RDF storage layouts, and [David et al., 2012] discusses available storage solutions for Semantic Web data (mainly RDF). [Bailey et al., 2005] introduces Web query languages and categorise them into three different groups, according to the format of the data they can retrieve: XML, RDF and Topic Maps. [Schmidt et al., 2009; Bizer and Schultz, 2009; Butt

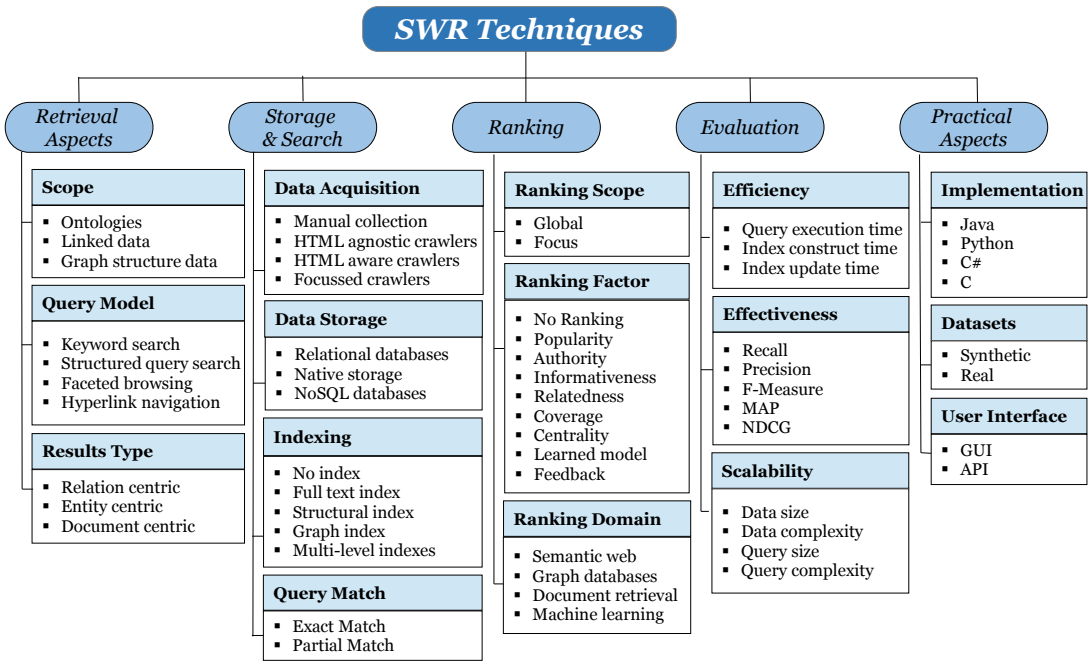


Figure 4.1: The dimensions used to characterize Semantic Web retrieval techniques

and Khan, 2014] evaluated the performance of state-of-the-art Triple stores' query engines. These works have designed query sets to check the performance of RDF triple stores for various SPARQL query language features. Another survey by [Noy and d'Aquin, 2012] presents a detailed analysis of ontology libraries but does not include several ontologies and linked data retrieval techniques.

While all these surveys analyse and compare the techniques for a particular aspect, we aim to develop a taxonomy that characterises all aspects of SWR and to provide a comprehensive analysis of current approaches to SWR along this taxonomy. Illustrating the gaps in current approaches to SWR will help to identify future research directions for SWR.

4.2 A Taxonomy of Semantic Web data Retrieval Techniques

In this section, we describe a taxonomy for SWR techniques. Our aim in developing this taxonomy is to provide a clearer picture of current approaches to retrieve Semantic Web data, and to identify gaps in these techniques which will help us to identify future research directions. We describe 16 dimensions of these techniques which we categorised into five main topics, as illustrated in Figure 4.1. In the following sections, we discuss each dimension in detail, while we also present an overview of the methodologies or techniques applied in these dimensions.

4.2.1 Retrieval aspects

All existing SWR techniques can be categorized into three major dimensions with respect to the retrieval design decisions: the type(s) of the data that can be explored with the approach, the way(s) a user can initiate the retrieval process, and the type(s) of the output as a result of a user's query.

4.2.1.1 Retrieval Scope

SWR techniques can be classified into those that explore schemata defined by ontologies describing a conceptualization for a domain of interest and those that explore data generated according to these schemata. The former are referred as '*ontology retrieval techniques*' and the latter as '*linked-data retrieval techniques*'. The linked-data-retrieval techniques [Hogan et al., 2011; Oren et al., 2008; Tummarello et al., 2010] focus on the retrieval of entities, relationships among entities, and sub-graphs. While the ontology-retrieval techniques [Alani et al., 2006; Ding et al., 2004; Vandenbussche and Vatant, 2014; Butt et al., 2014b] find the classes and properties within or across ontologies, and ontologies themselves. Both these type of approaches focus on different components of the retrieval process. The large size of linked data available requires retrieval techniques to mainly focus on efficient indexing and query evaluation plans. On the contrary, datasets that only consist of ontologies are relatively small and thus the ranking of results is more relevant in the retrieval process than the indexing and efficient query plan execution. '*Graph-retrieval techniques*' [He et al., 2007; Yang et al., 2014] is a category of SWR techniques comprised of the approaches proposed for general graph-based data but which are also applicable to and/or evaluated on the Semantic Web data retrieval task.

4.2.1.2 Query Model

SWR techniques generally consider one or more out of four query models: '*keyword search*', '*structured query search*', '*faceted browsing*',and '*hyperlink-based navigation*'. In keyword-based SWR techniques [Ding et al., 2004; He et al., 2007], a user poses a query string composed of one or more keywords, while the results are retrieved based on a match to one or more keywords in the query string. Structured query search introduces complexity while providing more flexibility to meet the user's requirements by retrieving results for a user specified pattern. Most of the SWR techniques [d'Aquin and Motta, 2011; Vandenbussche and Vatant, 2014; Wu et al., 2013; Yan et al., 2005; Yuan and Mitra, 2013] provide just an endpoint to query the data through the SPARQL structured query languages or graph queries. Few techniques [Noy et al., 2009; Oren et al., 2006] allow the user to find and filter the results based on a faceted browsing approach. Each facet is a characteristic (i.e. property), and the facet values are object values for that characteristic. Facets can be fixed irrespective of the search result as defined by the UI developer or generated dynamically based on the characteristics of the search results [Oren et al., 2006]. Hyperlink-based techniques [d'Aquin and Motta, 2011; Smith et al., 2007; Oren et al., 2008] facili-

tate users to navigate within the data. Each hyperlink is a predefined query that is executed when a user clicks on it. Keyword-search, faceted browsing and hyperlink-based navigation facilitate naïve users in exploring data, whereas structured query interfaces are for expert users; they need to know the syntax of the query language and the underlying schema of the data.

4.2.1.3 Result Type

The examined SWR approaches mostly consider one of three different output types to facilitate users in the exploration of the data. (i) *Document-centric* approaches [Oren et al., 2008; Butt et al., 2014b] list URIs or labels of matched documents (i.e. ontologies) and/or document parts (i.e. classes, properties and entities). The document-centric approaches may list URI's of same ontologies (resp. resources) multiple times containing different pieces of information about ontologies (resp. resources). (ii) *Entity-centric* approaches consolidate available data about the entity from multiple documents and the consolidated information is presented as a profile of the entity [Hogan et al., 2011; Tummarello et al., 2010]. Therefore, rather than listing matched documents as incomplete pieces of information, an entity-centric search outputs one or more matched entities with their available profile in the dataset. (iii) *Relation-centric* approaches [Anyanwu et al., 2005; Cheng et al., 2014] find relationships between entities. Mostly, structured queries or faceted browsing helps to perform relation-centric retrieval.

4.2.2 Storage and search approaches

4.2.2.1 Data Acquisition

The quality of a retrieval system depends on the quality of the underlying dataset. Data collection is mostly done in two ways: (1) *manual collection* – an admin or an owner collects a dataset manually considering the requirement or scope of the designed approach, and (2) *linked data crawler* – an application that gathers a collection of linked data as quickly and efficiently as possible. Existing linked data crawlers can be divided into three categories based on their crawling approach: (i) *HTML agnostic crawlers* do not crawl HTML documents. Therefore, these crawlers [Hogan et al., 2011] are not able to discover linked data embedded in HTML documents and RDF documents surrounded by HTML documents. (ii) *HTML aware crawlers* crawl both RDF and HTML documents and follow RDF and HTML links within them. However, when crawling, the crawler visits many HTML documents that have no embedded linked data and do not point to any RDF documents. (iii) *Focused crawlers* use a limited HTML crawling approach to control the efficiency of HTML crawling. These crawlers crawl both RDF and HTML documents but limit the crawling space for HTML documents. For example, [Oren et al., 2008] crawls only those HTML documents that are explicitly provided as endpoints by users and extracts embedded linked data and 'href' links with '.rdf' extension within them.

4.2.2.2 Storage

SWR techniques generally consider one of three storage structures: (1) *Native Storage:* SWR approaches [Hogan et al., 2011; He et al., 2007] deploy persistent storage with their designed storage architecture and are generally considered to be more efficient than the ones relying on relational databases [Butt and Khan, 2014]. (2) *NoSQL Databases:* Some of the SWR techniques use NoSQL databases to increase processing power and storage. Hadoop[1] is one of the most widely used NoSQL databases, used for example in Sindice [Oren et al., 2008]. (3) *Relational Databases:* SWR approaches employ traditional relational database management systems such as Microsoft SQL[2], MySQL[3], and Oracle[4] to store triples or quads. Semantic Web data is stored in a *vertical representation* - a big triple table or quad table, or in a *horizontal representation* - property tables and vertical partitioning. This storage approach was mainly adopted by approaches [Ding et al., 2004] in the early days of the Semantic Web, but due to the slow response time its no longer a choice.

4.2.2.3 Indexing

Various techniques for indexing linked data have been developed since the advent of the Semantic Web; several surveys of these techniques have been presented [Luo et al., 2012]. In this work, we divide RDF data indexing into four broad categories. The investigated SWR techniques implement one or more types of these four indexes.

1. **Full-text Index:** is implemented as an inverted index composed of a lexicon, i.e., a dictionary of terms that allows fast term lookup; and of a set of inverted lists, one inverted list per term. However, compared to traditional document-based inverted indexes, the difference is in the structure of the inverted lists. Based on the structural difference in the inverted list full-text indexes in Semantic Web are further divided into *node-based full-text indexes* and *graph-based full-text indexes*. In node-based indexes (resp. graph-based indexes) the inverted lists are composed of a list of the resource/node identifiers (resp. ontologies identifiers) for each term in the lexicon. To improve the space and time complexity of full-text indexes, some SWR approaches separate node-based full-text indexes for entities, attributes and object values into an *entity-node inverted index*, *attribute-node inverted index*, and *value-node inverted index*.

2. **Structural Indexes:** are specially designed for RDF data stores [Harth et al., 2007]. Such indexes can be classified into those that index a triple (subject-predicate-object) and those that index a quadruple (context-subject-predicate-object). The former are known as *triple indexes* and the latter as *quad indexes*. In contrast to a separate index on subject, predicate, object and/or context where join operations are required to derive the answer to a query, a complete index of

[1] http://lucene.apache.org/hadoop
[2] http://www.microsoft.com/en-au/server-cloud/products/sql-server/
[3] http://www.mysql.com/
[4] http://www.oracle.com/index.html

a quad or triple pattern allows a direct lookup on multiple dimensions without a join operation. To make the search more efficient, indexes with all possible patterns for a quadruple or a triple are implemented, i.e. $4^2 = 16$ and $3^2 = 9$ indexes for quadruple and triple respectively.

3. **Graph Indexes:** Recently, graph indexes have been introduced to support efficient structural queries over graph or RDF data. Compared to traditional indexes where each node has a key-value pair in the index, the difference is in the content structure of the key-value pair in the graph index. Traditionally, the key in an index node is either a text or an identifier; in graph indexes, a key is a subgraph (patterns), and its value is a set of database graphs (ontologies) that contain the subgraph. Data structures adopted to implement graph indexes to enhance the filtering include *feature-matrices* [Yan et al., 2005], *graphs* [Yang et al., 2014; Wu et al., 2013], and *lattices* [Yuan and Mitra, 2013].

4. **Multi-level Indexes:** Other than creating multiple types of indexes SWR techniques also introduce multi-level indexes to improve the efficiency of the retrieval process. One such approach is presented in [He et al., 2007]. Indexes at different levels narrow down the search space by reducing the size of the relevant dataset to the query.

4.2.2.4 Query Match

The efficiency and effectiveness of the query evaluation are heavily influenced by how the matches are found in the data collection. The matched results for a query in a repository are found either for an *exact match* or for a *partial/approximate match*. The exact match is efficient since an exact keyword or structure query match always ensures the fewer results for a user; however, it sometimes results in an empty resultset if either an exact match is unavailable or the user is unaware of the contents or the structure of the dataset. On the other hand, a partial match enhances the chances to come up with approximate or similar results for the user, but with the disadvantage of a potentially large number of results that need some ranking mechanism to suggest the most appropriate result to the user.

4.2.3 Ranking

In addition to providing the information in response to a user query, some retrieval approaches rank the results. The ranking indicates which result (entity or ontology) is deemed to be the most appropriate for the query. The ranking models designed or adapted for Semantic Web data ranking can be distinguished along several dimensions; some of them are discussed in this section.

4.2.3.1 Ranking Scope

Ranking scope denotes if a SWR technique is query dependent or not. The query dependent approaches are referred to as *'focused-ranking'*- i.e. the ranking model is

applied only to the result set, and the relative order of each result in the result set is computed. The second class of ranking approaches which we refer to as '*global-ranking*' are implemented on the complete dataset (ontologies or linked data) irrespective of the query. Since the focused-ranking approaches are applied only on a subset (results) of the dataset they lead to a higher efficiency in computing the ranks; however, the ranks calculated are not the global optimum. Global ranking is more time consuming but produce globally optimum ranking scores of query results.

4.2.3.2 Ranking Factor

One other important dimension of ranking is the 'ranking factor' based on which the ranks are calculated. The factors that have been used in different ranking approaches are explained here:

1. **Popularity:** Similar to the document retrieval domain most of the ranking techniques adopted for Semantic Web data order the output of a user query concerning the popularity of a result in a dataset. Different SWR techniques have adopted different popularity measure models, originally designed for information retrieval. PageRank [Page et al., 1998] and TF-IDF [Salton and Buckley, 1988] are the most widely used popularity measures for Semantic Web data ranking.

2. **Authority:** Authority, a measure of *trustworthiness*, is another factor by which individual resources or documents (ontologies) are ranked. HITS [Kleinberg, 1999], designed for informational retrieval, is used to compute the authority of the resources in [Butt et al., 2014b]; and variations of HITS are also investigated in [Hogan et al., 2006].

3. **Informativeness:** For Semantic Web data, informativeness is a measure of the degree of information carried by each resource that helps to identify it. Several SWR techniques [Meymandpour and Davis, 2013; Cheng et al., 2011] adopted Shannon entropy [Shannon, 2001] as an informativeness measure, according to which informativeness of a resource is the negative log of the probability of the presence of the resource in a given dataset.

4. **Relatedness:** Relatedness is the similarity between features (property-value pair) of a resource. A resource is ranked higher if features of the resource are related to each other. Different relatedness models have been proposed such as WordNet to measure the relatedness between two features based on their text similarity, or *distributional relatedness*, i.e., two features are more related if they more often co-occur in a certain graph (ontology).

5. **Coverage:** Coverage is a query-dependent ranking factor that measures how much of a query term or a structured query is covered by a resource. The Vector Space Model (VSM) [Salton et al., 1975] and BM25 [Robertson et al., 1995] are document retrieval models that compare the similarity between a query and the

matched document. These models have been adapted to the task of resources and ontology ranking.

6. **Learning a model:** Other approaches for ranking Semantic Web data are rooted in 'learning to rank', a technique developed for machine learning [Trotman, 2005]. In these approaches, different graph/ontology features are selected (or computed) and by these features a ranking model is learnt and then the learned model is used to produce the ranking for search results [Butt et al., 2016].

7. **Centrality:** Some ranking models designed or adopted by SWR approaches consider the centrality of a concept/resource to compute their ranks. Some approaches find the centrality as connectivity of a node/resource in a graph/ontology [Freeman, 1977]. Mostly, it is a measure of the number of relations or edges for a concept or a node.

8. **User Feedback:** Some SWR techniques [Noy et al., 2009] consider user feedback such as view count and query log to compute the ranking of the result-set.

4.2.3.3 Ranking Domain

SWR techniques are either designed purely for Semantic Web data or are borrowed from other domains. Most of the approaches are adopted from the '*document retrieval*' domain including Pagerank, HIT, VSM, TF-IDF and BM25. Because of the graph structure of the RDF model, many of the SWR techniques adopt ranking approaches that were designed for *graphs* in general, i.e. shortest path [Goldberg and Harrelson, 2005] and centrality measure [Freeman, 1977]. A recent trend for ranking Semantic Web data is the adaptation of learning-to-rank [Trotman, 2005] approaches from the *machine learning* domain. However, these models are not applicable to the Semantic Web data in its original form, because of the nature of the data. Therefore variants of these models are implemented, and some of them are studied in [Butt et al., 2014a].

4.2.4 Evaluation

The performance of a SWR technique needs to be evaluated regarding three factors: efficiency, effectiveness, and scalability. The efficiency of a SWR approach provides a measure of how fast the retrieval process is, while the effectiveness of the approach is measured by the accuracy of the retrieval model and quality of the retrieved results. Scalability measures the SWR technique for its capability to handle large-scale datasets and complex queries.

4.2.4.1 Efficiency

Efficiency is evaluated using measures that are dependent upon resource utilisation on the computing platform (i.e. memory consumption) or measures that are based on the time taken to retrieve the relevant results. Existing approaches evaluate the time

taken on different processes of the retrieval process including: (1) *query evaluation time*, (2) *Index construction time*, and (3) *Index update time*.

4.2.4.2 Effectiveness

One or more out of five popular metrics are used to evaluate the effectiveness of an SWR approach.

1. **Recall:** a fraction of *relevant* documents that are *retrieved* i.e.

$$\begin{aligned} Recall &= \frac{\#(relevant - results - retrieved)}{\#(relevant - results)} \\ &= \frac{retrieved}{relevant} \end{aligned}$$

2. **Precision:** a fraction of *retrieved* results that are *relevant*

$$\begin{aligned} Precision &= \frac{\#(relevant - results - retrieved)}{\#(retrieved - results)} \\ &= \frac{relevant}{retrieved} \end{aligned}$$

   It is hard to determine the relevance and irrelevance of all results for queries resulting in a larger number of matched results; therefore mostly precision is determined for a cut-off value i.e. for top-k results. Precision at k (P@k) for a k value is calculated as:

$$p@k = \frac{\#\ relevant\ results\ in\ top\ k\ results}{k}$$

3. **F-Measures:** F-Measure is a measure that trades off precision versus recall which is the weighted harmonic mean of precision and recall, i.e.,

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

4. **Mean Average Precision:** The average precision for the query Q of a SWR technique is defined as

$$AP(Q) = \frac{\sum_{i=1}^{k} rel(r_i) * P@i}{k}$$

   where *rel(r_i)* is 1 if r_i is a relevant resource for the query Q and 0 otherwise, *P@i* is the precision at i and k is the cut off value. *MAP* is defined as the mean of *AP* over all queries run in an experiment and is calculated as:

$$MAP = \frac{\sum_{Q \in \mathbb{Q}} AP(Q)}{|\mathbb{Q}|}.$$

5. **Normalize Discounted Cumulative Gain (NDCG):** NDCG is a standard evaluation measure for ranking tasks with a non-binary relevance judgement. NDCG is defined based on a gain vector G, that is, a vector containing the relevance assessments at each rank. Then, the discounted cumulative gain measures the overall gain obtained by reaching rank k, putting more weight at the top of the ranking.

$$DCG(Q) = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(1+i)}.$$

The NDCG is computed by dividing DCG by its optimal value iDCG which puts the most relevant results first. iDCG is calculated by computing the optimal gain vector for an ideal ordering.

4.2.4.3 Scalability

SWR techniques can be evaluated using measures that are dependent on the size (no. of triples) and the structural complexity of the dataset and the query, and/or on the flexibility of the approach. The former is referred to as *space scalability*, and the latter is referred to as *structural scalability*. Evaluations are conducted to compute resources utilisation (including memory and time). For a scalable approach, the resources usage does not grow to intolerable levels as the size or complexity of the data set or query increases. The metrics for scalability are characterised as: (i) *data size*, (ii) *data complexity*, (iii) *query size* and (iv) *query complexity*.

4.2.5 Practical aspects

The final category covers practical aspects of SWR techniques, including the type of datasets used for implementation or experimental evaluations, how the solution was implemented, and the kind of user interface(s) provided to use these techniques.

4.2.5.1 Implementation

This dimension specifies the implementation techniques that have been used to implement or to prototype a SWR technique to conduct its experimental evaluation. Some solutions proposed in the literature provide only logical proofs but they have not been evaluated experimentally, or no details about their implementation have been published.

4.2.5.2 Dataset

Experimental evaluation on one or ideally several datasets is important for the critical evaluation of a SWR technique. Due to the difficulty in obtaining real-world data that contain a large number of triples, synthetically generated datasets are commonly used.

4.2.5.3 User Interface

Most of the SWR techniques are developed primarily for interactive browsing while additionally providing programmatic access to its content. For browsing *Graphical user interfaces* (mostly Web-forms) are designed to make it a more interactive experience for the user, while programmatic access is made available through *Web services* that enable application developers to use the content of the SWR techniques in their application.

4.3 Discussion

In this section, we analyse the surveyed SWR techniques as characterised in Table 4.1 with regard to the proposed taxonomy. These SWR techniques were described in detail in Chapter 3. This analysis highlights several areas of potential future research directions in SWR. Since the beginning of the development of techniques that aim to provide solutions for SWR, there is a clear path of progress, starting from early techniques that solve the problem of Semantic Web data retrieval for exact keywords document search using naïve approaches, moving on to entity search techniques that allow advanced faceted browsing. Still, some research gaps can be focused on in the future.

- **Dynamic Faceted Browsing:** Most of the ontology search engines and libraries either do not facilitate faceted browsing at all or filter results based on fixed facets for all searches (e.g. LOV [Vandenbussche and Vatant, 2014] and BioPortal [Noy et al., 2009]). A more satisfying approach seems to lie in finding facets dynamically based on the matched results for a query. However, a major hurdle in identifying the dynamic facets is the syntactic diversity in describing the same property, for example, a *title* of a resource can be described as a *name*, a *title*, a *label* etc. in different vocabularies. A potential solution might be in clustering similar types of properties into a single group using machine learning and data mining techniques and declaring the group of properties as a facet rather than having individual properties as facets.

- **Ontology Retrieval:** Most of the ontology search systems retrieve ontological terms (concepts and relations), and some provide ontology search based on some keywords. The ontology search systems that retrieve matched ontologies for multi-keyword queries often returns ontologies that match to one of the query terms. However, they lack a criterion to find the relevant ontologies

that cover most of the query terms or related concepts to these terms. BioPortal [Noy et al., 2009] provides an opportunity to find an ontology based on its text description, however, it is a domain dependent ontology library and does not deal with all type of ontologies. A general solution for ontology retrieval based on text descriptions or several keywords still needs to be devised. This shortcoming leads to **RQ4** of this thesis, i.e., "How to find the most relevant resources and ontologies that cover one or more resources users are interested in?" To address this research question, we propose an ontology retrieval framework for keyword queries; our results and findings are presented in Chapter 8.

- **Ontology Ranking Models:** Ontology collections are limited in size, therefore ranking becomes the core task for ontology search engines and libraries, rather than efficient search. However, ontology ranking is challenging, because search results are a match for a search term with a more expressive class, property or ontology description. There may exist many ontologies that contain concepts and relations with their labels matching the keyword query, however, they have been described differently mainly concerning their: (i) *perspective* - A concept may be defined in different perspectives, e.g., a person class is defined in many ontologies, for example, the 'foaf' ontology captures the social aspects of person, whereas the 'appearance' ontology models the natural attributes of a person, i.e. weight, height, and nature, (ii) *levels of detail* - the concepts are defined in the same perspective in different ontologies but different levels of detail, i.e. abstract or detailed, and (iii) *extension* - the concepts are defined in one ontology and then extended in another ontology. The problem is how to find and order many matched results for a keyword search to satisfy a user's information need. Most of the ontology retrieval systems do not focus on ranking at all [Noy and d'Aquin, 2012] and others adopted ranking approaches that are rooted in a graph or document retrieval ranking model without considering the underlying nature of ontologies. This provides ample opportunities for research to significantly improve the ranking of ontologies or ontological terms based on a more expressive user query. The challenge establishes two of the research questions i.e., **RQ3:** How to rank relevant resources and ontologies for keyword queries? and **RQ5:** What are the inherent characteristics of ontologies that can be used to rank relevant resources and ontologies for keyword queries? To address these research questions, we work extensively on ranking models for ontology retrieval. The thesis proposes concept ranking models in Chapter 6 and 7, and ontology ranking model in Chapter 8.

- **Linked data retrieval effectiveness vs. efficiency:** linked data retrieval approaches can be classified into two major categories: (i) *Effectiveness oriented techniques* - which apply ranking models to retrieve the most appropriate answers, i.e. [Maedche et al., 2001; Hogan et al., 2006; Oren et al., 2008] (ii) *Efficiency oriented techniques* - which mainly focus on efficient indexing to achieve the efficiency in retrieving results with less focus on ranking, i.e. [He et al., 2007; Wu et al., 2013; Yang et al., 2014]. There is scope for linked data re-

trieval techniques that make a reasonable trade-off between effectiveness and efficiency of the retrieval approaches.

- **Ranking of triples for entity retrieval:** In recent years the linked data retrieval paradigm is shifting from document retrieval to entity retrieval [Hogan et al., 2011; Oren et al., 2008]. The entity retrieval process finds entities and consolidates attributes for an entity from multiple data sources. It requires a ranking of triples for the entity to prioritise relevant attributes of that entity. Existing approaches rank properties in a general context based on their occurrence in a dataset. However, the ranking of a property depends on the entity it belongs to. The property may be attached to more than one entity and the relative importance of the property will vary for each entity. Secondly, the object values for multivalued properties mostly have different ranking criteria depending upon the entity to which the property belongs to, but they are also ranked according to its popularity in current approaches. This constitutes a significant gap between the state-of-the-art entity ranking techniques and the ideal ranking and presents opportunities for future research.

- **An evaluation framework for Semantic Web data retrieval techniques:** There is currently no comprehensive evaluation strategy that facilitates the comparative evaluation of different SWR techniques with regards to their effectiveness, efficiency and scalability. Researchers have used a variety of evaluation measures and datasets (both real and synthetic), which makes comparing existing techniques difficult. It is currently not possible to determine which technique(s) perform better than others on data with different characteristics and of different sizes. So far, it seems that no single SWR technique has outperformed all other techniques in all aspects on large datasets. A benchmark on ontology ranking [Butt et al., 2014a] has been published recently. It contributes an ontology collection, ten benchmark queries, a gold standard and evaluation of eight state-of-art ranking model on the task of ontology search. However, the benchmarks deal with ontology concepts ranking only. No comprehensive study compares many existing techniques within the same framework and on different datasets. Conducting such extensive experimental studies is one avenue of research that would be highly beneficial to understand the characteristics of these techniques better. This shortcoming is underpinning the research question **RQ6:** How to evaluate the newly emerging ontology libraries and search engines in comparison to existing ones? We address this in Chapter 5 of the thesis by proposing an ontology evaluation framework.

4.4 Summary

We have identified sixteen dimensions that allowed us to characterise Semantic Web data retrieval techniques, and to generate a taxonomy of such techniques. This proposed taxonomy can be used as a comparison and analysis tool for Semantic Web

data retrieval techniques. Through this taxonomy, we have identified various shortcomings of current approaches that suggest several future research directions in this field. We will address some of these identified shortcomings in Chapters 5, 6, 7 and 8 of this thesis.

Table 4.1: Categorization of Prominent SWR Techniques

	Search Aspect			Storage/Search				Ranking			Evaluation			Practical Aspects		
Techniques	Search Scope	Query Model	Result Type	Data Aquisition	Data Storage	Indexing	Query Match	Arena	Factor	Domain	Efficiency	Effectiveness	Scalability	Implementation	Datasets	User Interface
OntoSelect04	Ontology	Keyword Faceted	Document	HTML Agnostic, Manual	-	-	-	Global	Popularity	Document	-	-	-	-	Real	GUI, API
OLS06	Ontology	Keyword, Navigational	Document	Manual	-	-	-	-	-	-	-	-	-	-	Real	GUI, API
OBO07	Ontology	Navigation	Document	Manual	-	No-Index	-	-	-	-	-	-	-	-	Real	GUI
OntoSearch07	Ontology	Keyword Faceted	Document	Manual	Native	Structure	Partial	Global	Coverage Popularity Relatedness	Document	QueryEx	-	DataSize	-	Real	GUI
oeGov09	Ontology	Faceted	Document	Manual	Native	No-Index	-	Global	-	-	-	-	-	-	-	API GUI
BioPortal09	Ontology	Keyword Faceted	Document	Manual	-	-	Exact	Focus	Feedback	Document	-	-	-	Java	Real	GUI API
Cupboard09	Ontology	Keyword	Document	Manual	-	-	-	-	-	-	-	-	-	-	-	-
TONES10	Ontology	Browsing	Document	Manual	-	-	-	-	-	-	-	-	-	-	Real	API
NCBO10	Ontology	Keyword	Document	Manual	-	-	Partial	Global	Centrality Coverage	Document	-	Precision	-	-	Real	API
LOV14	Ontology	Keyword Structure	Document	Manual	Native	FullText	Exact	Global	Popularity	Document	-	-	-	Java	Real	GUI API
BiOSS14	Ontology	Keyword	Document	Manual	Relation	-	Partial	Global	Coverage Relatedness Popularity	Graph Document	QueryEx	Precision	-	-	Real	GUI API
OntoKhoj03	Ontology	Keyword	Document	HTML Agnostic	Relation	FullText	Partial	Global	Popularity	Document	-	F-Measure	-	-	Real	GUI API
Swoogle04	Ontology Linked	Keyword	Document	HTML Agnostic	Relation	FullText	Partial	Global	Popularity	Document	QueryEx	-	-	Java	Real	GUI
WATSON11	Ontology Linked	Keyword Structure	Document	HTML Agnostic	Native	FullText Structure	Exact	-	NoRanking	-	-	-	-	Java	Real	GUI API
AKTiveRank06	Ontology	Keyword	Document	-	-	-	Partial	Focus	Popularity Coverage Centrality	Graph Document	-	Precision	-	-	Real	-
SAB06	Ontology	Browsing	Document	Manual	Relation	FullText	Partial	Global	Coverage	Document	-	-	-	-	Real	-

Table 4.1: Categorization of Prominent SWR Techniques

	Search Aspect			Storage/Search				Ranking			Evaluation			Practical Aspects		
Techniques	Search Scope	Query Model	Result Type	Data Aquisition	Data Storage	Indexing	Query Match	Arena	Factor	Domain	Efficiency	Effectiveness	Scalability	Implementation	Datasets	User Interface
WebCORE07	Ontology	Keyword	Document	Manual	-	-	Partial	Global	Coverage	Docuemnt	QueryEx	Precision	-	-	Real	GUI
OntoQA07	Ontology	Keyword	Document	Manual	-	-	Partial	Global	Connectivity	Document	-	Precision	-	-	Real	-
Sindice08	Linked	Keyword	Document	Focussed	NoSQL	FullText	Partial	Focus	Coverage	Document	QueryEx IndCrt IndUpt	-	DataSize	Java	Real	GUI API
Sig.ma10	Linked	Keyword	Entity	-	NoSQL	FullText	Partial	HTML aware	Authority	Document Coverage	QueryEx	-	-	Java	Real	GUI API
SWSE11	Linked	Keyword	Entity	HTML Agnostic	Native	Structure	Partial	Focus	Authority Popularity	Document	QueryEx IndCrt IndUpt	-	DataSize	Java	Real	GUI
SemRank05	Linked	-	Relation	Manual	Relation	Structure	Exact	Focus	Informative	Document	-	-	-	-	Synthetic	-
ReConRank06	Linked	Keyword	Document	Manual	Native	Structure	Partial	Focus	Authority Popularity	Document	-	-	DataSize	Java	Real	
RSS08	Linked	Keyword	Document	Manual	Native	Structure	Partial	Global, Focus	Popularity	Document	-	Precision	-	-	Real	-
TripleRank09	Linked	Faceted	Relation	Manual	Native	Structure	-	Global	Authority, Cenrality	-	-	Precision	-	-	Real	-
Harth09	Linked	-	Entity	Manual	Native	Structure	-	Focus	Popularity	Document	QueryEx	-	-	-	Real	-
DING10	Linked	Keyword	Entity	HTML Aware	NoSQL	Structure	Partial	Focus	Popularity	Document	QueryEx.	Precision	-	-	Real	-
Dali12	Linked	-	Entity	Manual	Native	Structure	-	Global	Learned	Learning	-	NDCG	-	-	Real	-
BLINK07	Graph	Keyword	Document	Manual	Native	MultiLevel	Partial	Focus	Centrality	Graph	QueryEx IndexCrt	-	DataSize	-	Real	-
Lindex11	Graph	Structured	Document	Manual	Native	Graph MultiLevel	Partial	-	-	-	QueryEx IndCrt IndUpt	-	DataSize	-	Synthetic	-
Wu13	Graph	Structured	Document	Manual	Native	Graph	Partial	-	-	-	QueryExec IndCrt	Precision MAP,NDCG	DataSize	-	Real Synthetic	-
NeMa13	Graph	Structured	Document	Manual	Native	Graph	Partial	-	-	-	QueryEx IndexCrt IndexUpt	Precision Recall F-Measure	DataSize	-	Real	-
SLQ 14	Graph	Keyword Faceted	Document	Manual	Native	Structure FullText	Partial	Focus	Centrality	Learning	QueryEx	Precision MAP,NDCG	DataSize	Java	Real Synthetic	-

Chapter 5

Evaluation Benchmark

This chapter addresses two of the research questions: **RQ6** i.e., how to evaluate the newly emerging ontology libraries and search engines in comparison to existing ones? and **RQ2** i.e, do the document ranking models suffice for ontology ranking? In this chapter, we present a benchmark to evaluate the effectiveness of ontology ranking models. We begin with a discussion of the ranking algorithms that we have implemented for benchmark experiments in Section 5.2. In Section 5.3, we describe the evaluation setup. We then present the results and a result analysis in Section 5.4. Section 5.5 discusses some recommendations for the improvement of the ranking models for ontology search, before we conclude in Section 5.6.

5.1 An introduction to CBRBench

The task of ranking resources defined in ontologies can be based on many different criteria [Gangemi et al., 2005], for example, how well an ontology meets the requirements of certain evaluation tests (e.g. [Guarino and Welty, 2002]) or on methods to evaluate general properties of an ontology based on some requirement (e.g. [Lozano-Tello and Gomez-Perez, 2004]). However, only limited work has been proposed, by the Semantic Web community, to rank the returned resources based on a user posted keyword query such that the most relevant results appear higher on the list. [Alani et al., 2006] propose four measures (i.e. Semantic Similarity, Betweenness, Density and Class Match Measure) to evaluate different representational aspects of the ontology and calculate its ranking. Moreover, in the information retrieval community many algorithms, such as the vector space model, the boolean model, BM25, tf-idf, etc. have been proposed to identify and rank a small number of potentially relevant documents through a top-*k* document retrieval. The Semantic Web community has adopted these models for ranking ontologies and ontological resources. To the best of our knowledge, no systematic study has been conducted to compare the performance of these state-of-the-art ranking techniques on the task of ranking resources in ontologies. For our study, we have implemented eight ranking algorithms, four of which have been proposed by the information retrieval community whereas the others were adapted for the ranking of ontologies by [Alani et al., 2006]. We defined a set of queries derived from a real query log and computed the ranking for these

queries on a collection of ontology resources that we have crawled with a seed set of ontology URIs derived from prefix.cc. We computed a baseline ranking and established a ground truth by asking ten ontology engineers to manually rank ontologies based on a given search term from the collection of resources obtained by the baseline ranking. We compared the ground truth derived through the human evaluation with the results from each of the ranking algorithms. We calculated the precision at k, the mean average precision and the discounted cumulative gain of the ranking algorithms in comparison to a ground truth to determine the best model for the task of ranking resources/ontologies. The contribution of this work are:

- a design of a benchmark suite named CBRBench, for Canberra Ontology Ranking Benchmark, including an ontology collection, a set of queries and a ground truth established by human experts for evaluating ontology ranking algorithms,
- a methodology for resource ranking evaluation where we discuss many of the decisions that need to be made when designing a search evaluation framework for resources defined in ontologies,
- the evaluation of eight ranking algorithms through these benchmarks, and
- a set of recommendations derived from an analysis of our experiment that we believe can significantly improve the performance of the ranking models.

5.2 Baseline Ranking Models

We have chosen eight different ranking models that are commonly used for ranking documents and/or ontologies and applied them to the task of ranking resources/ontologies according to their relevance to a query term. These eight ranking models can be grouped into two different categories.

1. **Content-based Ranking Models**: tf-idf, BM25, Vector Space Model and Class Match Measure.
2. **Graph-based Ranking Models**: PageRank, Density Measure, Semantic Similarity Measure and Betweenness Measure.

Because of the inherent graph structure of ontologies, graph-based ranking models can be used for ranking as such. However, content-based ranking models (e.g. tf-idf, BM25 and Vector Space Model) need to be tailored towards ontologies so that instead of using a word as the basic unit for measuring, a resource r in an ontology is considered as the measuring unit. Therefore, the relevance of a query word to the ontology is the sum of the relevance of all the resources that match the query term. For tf-idf we compute the relevance score of the resource, all other algorithms generate a cumulative relevance score for the ontology and resources are ranked according to the relevance score of their corresponding ontology. The matched resource set for each term/word is selected from a corpus if a word exists in the value of the

Table 5.1: Notation used

Variable	Description
$\mathbb{O}$	Corpus: The ontology repository
N	Number of ontologies in $\mathbb{O}$
O	An ontology: $O \in \mathbb{O}$
r	A resource uri: $r \in O$ & $r \in URI$
z	Number of resources in O
Q	Query String
q_i	query term i of Q
n	number of keywords in Q
σ_O	set of matched uris for Q in O
$\sigma_O(q_i)$	set of matched uris for q_i in O : $\forall\ r_i \in \sigma_O$, $r_i \in O$ & r_i matches q_i
m	number of uris in $\sigma_O(q_i)$

1) `rdfs:label` 2) `rdfs:comment`, or 3) `rdfs:description` property of that resource or if the word is part of the URI of the resource. As most of the existing adaptations of graph ranking models for ontology ranking do not compute a ranking for properties in an ontology, we only consider the ranking of classes/concepts in this study. However, it turns out that only 2.6% of all resources in our corpus (cf. Section 5.3) are properties.

In the following sections, we introduce all ranking models and describe the choices we made to adapt them for the ranking of resources in ontologies. Common notations used in the following sections are shown in Table 5.1.

5.2.1 tf-idf

Term frequency inverse document frequency (tf-idf) [Salton and Buckley, 1988] is an information retrieval statistic that reflects the importance of a word to a document in a collection or corpus. For ranking ontologies we compute the importance of each resource r to an ontology O in an ontology repository, where $r \in R$: R = URI only (i.e. excluding blank nodes and literals).

$$\begin{aligned} tf(r,O) &= 0.5 + \frac{0.5 * f(r,O)}{max\{f(r_i,O) : r_i \in O\}} \\ idf(r,\mathbb{O}) &= log\frac{N}{|\{O \in \mathbb{O} : r \in O\}|} \\ tf-idf(r,O,\mathbb{O}) &= tf(r,O) * idf(r,O) \end{aligned} \tag{5.1}$$

Here *tf(r,O)* is the term frequency for resource r in O. *tf(r,O)* is the frequency of r (number of times r appears in O) divided by the maximum frequency of any resource r_i in O. The inverse document frequency $idf(r,\mathbb{O})$ is a measure of commonality of a

resource across the corpus. It is obtained by dividing the total number of ontologies in the corpus by the number of documents containing the resource r, and then computing the logarithm of that quotient. The final score of r for this query is the tf-idf value of r.

$$score(r, Q) = tf - idf(r, O, \mathbb{O}) : \forall r\{\exists q_i \in Q : r \in \sigma(q_i)\} \tag{5.2}$$

5.2.2 BM25

BM25 [Robertson et al., 1995] is a ranking function for document retrieval used to rank matching documents according to their relevance to a given search query. Given a query Q, containing keywords q_1, ..., q_n, the BM25 score of a document d is computed by:

$$score(d, Q) = \sum_{i=1}^{n} idf(q_i, d) \frac{tf(q_i, d) * k + 1}{tf(q_i, d) + k * (1 - b + b * (\frac{|d|}{avgdl}))} \tag{5.3}$$

where $tf(q_i, d)$ is the term frequency and $idf(q_i, d)$ is the inverse document frequency of the word q_i. | d | is the length of the document d in words, and $avgdl$ is the average document length in the text collection from which the documents are drawn. k_1 and b are free parameters, usually chosen, in absence of an advanced optimisation, as $k_1 \in$ [1.2,2.0] and b = 0.75.

In order to tailor this statistic for ontology ranking we compute the sum of the score of each $r_j \in \sigma_O(q_i)$ for each query term q_i rather than computing the score for q_i. For the current implementation we used k_1 = 2.0, b = 0.75 and | O | = total number of terms (i.e. 3 * | axioms |) in the ontology. The final score of the ontology is computed as:

$$score(O, Q) = \sum_{i=1}^{n} \sum_{\forall r_j : r_j \in \sigma_O(q_i)} idf(r_j, \mathbb{O}) \frac{tf(r_j, O) * k + 1}{tf(r_j, O) + k * (1 - b + b * (\frac{|O|}{avgol}))} \tag{5.4}$$

5.2.3 Vector Space Model

The vector space model (VSM) [Salton et al., 1975] is based on the assumptions of the document similarities theory where the query and documents are represented as the same kind of vector. The ranking of a document to a query is calculated by comparing the deviation of angles between each document vector and the original query vector. Thus, the similarity of a document to a query is computed as under:

$$sim(d, Q) = \frac{\sum_{i=1}^{n} w(q_i, d) * w(q_i, Q)}{|d| * |Q|} \tag{5.5}$$

Here $w(q_i, d)$ and $w(q_i, Q)$ are weights of q_i in document d and query Q respectively. $|d|$ is the document norm and $|Q|$ is the query norm. For this implementation, we are considering the *tf-idf* values of a query term as weights. Therefore, the similarity of an ontology to query Q is computed as:

$$\begin{aligned} sim(O,Q) &= \frac{\sum_{i=1}^{n} tf - Idf(q_i,O) * tf - idf(q_i,Q)}{|O| * |Q|} \\ tf - idf(q_i,O) &= \sum_{j=1}^{m} tf - idf(r_j,O) : r_j \in \sigma_O(q_i) \\ tf - idf(q_i,Q) &= \frac{f(q_i,Q)}{max\{f(q,Q) : q \in Q\}} * log\frac{N}{|\{O \in \mathbb{O} : r \in O \& r \in \sigma_O(q_i)\}|} \\ |O| &= \sqrt{\sum_{i=1}^{z}(tf - idf(r_i,O))^2} \\ |Q| &= \sqrt{\sum_{i=1}^{n}(tf - idf(q_i,Q))^2} \end{aligned} \tag{5.6}$$

5.2.4 Class Match Measure

The Class Match Measure (CMM) [Alani et al., 2006] evaluates the coverage of an ontology for the given search terms. It looks for classes in each ontology that have matching URIs for a search term either exactly (class label 'identical to' search term) or partially (class label 'contains' the search term). An ontology that covers all search terms will score higher than others, and exact matches are regarded as better than partial matches. The score for an ontology is computed as:

$$score_{_{CMM}}(O,Q) = \alpha score_{_{EMM}}(O,Q) + \beta score_{_{PMM}}(O,Q) \tag{5.7}$$

where $score_{_{CMM}}(O,Q)$, $score_{_{EMM}}(O,Q)$ and $score_{_{PMM}}(O,Q)$ are the scores for class match measure, exact match measure and partial match measure for the ontology O with respect to query Q, α and β are the exact matching and partial matching weight factors respectively. As exact matching is favoured over partial matching, therefore $\alpha > \beta$. For our experiments, we set $\alpha = 0.6$ and $\beta = 0.4$ (as proposed in the original paper [Alani et al., 2006]).

$$\begin{aligned} score_{_{EMM(O,Q)}} &= \sum_{i=1}^{n}\sum_{j=1}^{m} \varphi(r_j,q_i) : r_j \in \sigma_O(q_i) \\ \varphi(r_j,q_i) &= \begin{cases} 1 & \text{if label}(r_j) = q_i \\ 0 & \text{if label}(r_j) \neq q_i \end{cases} \end{aligned} \tag{5.8}$$

$$
\begin{aligned}
score_{_{PMM}}(O,Q) &= \sum_{i=1}^{n}\sum_{j=1}^{m}\psi(r_j,q_i) : r_j \in \sigma_O(q_i) \\
\psi(r_j,q_i) &= \begin{cases} 1 & \text{if label}(r_j) \text{ contains } q_i \\ 0 & \text{if label}(r_j) \text{ does not contain } q_i \end{cases}
\end{aligned} \tag{5.9}
$$

5.2.5 PageRank

PageRank (PR) [Page et al., 1998] is a hyperlink based iterative computation method for document ranking which takes as input a graph consisting of nodes and edges (i.e. ontologies as nodes and `owl:imports` properties as links in this implementation). In each successive iteration, the score of ontology o is determined as a summation of the PageRank score in the previous iteration of all the ontologies that link (imports) to ontology O divided by their number of outlinks (owl:imports properties). For the k_{th} iteration the rank of ontology O i.e. ($score_k(O)$) is given as under:

$$
\begin{aligned}
score_k(O) &= \frac{\sum_{j \in deadlinks(\mathbb{O})} score_{k-1}(j)}{n} + \sum_{i \in inlinks(O)} \frac{score_{k-1}(i)}{|outdegree(i)|} \\
score_k(O) &= d * score_k(O) + \frac{1-d}{n}
\end{aligned} \tag{5.10}
$$

Here $deadlinks(\mathbb{O})$ are ontologies in corpus $\mathbb{O}$ that have no outlinks, i.e. they never import any other ontology. All nodes are initialised with an equal score (i.e. $\frac{1}{n}$, where n is the total number of ontologies in $\mathbb{O}$ before the first iteration. In the experimental evaluation, we set the damping factor d equal to 0.85 (a common practice), and we introduced missing `owl:imports` links among ontologies based on reused resources.

5.2.6 Density Measure

Density Measure (DEM) [Alani et al., 2006] is intended to approximate the information content of classes and consequently the level of knowledge detail. This includes how well the concept is further specified (i.e. the number of subclasses), the number of properties associated with that concept, the number of siblings, etc. Here $score_{_{DEM}}(O,Q)$ is the density measure of ontology O for query Q. $\Theta(r_j,q_i)$ is the density measure for resource r_j and w is a weight factor set for each dimensionality i.e. sub classes = 1, super classes = 0.25, relations = 0.5 and siblings = 0.5 and $k = n * m$ (i.e. number of matched r) for query Q.

$$
\begin{aligned}
score_{_{DEM}}(O,Q) &= \frac{1}{k}\sum_{i=1}^{n}\sum_{j=1}^{m}\Theta(r_j) : r_j \in \sigma_O(q_i) \\
\Theta(r_j) &= \sum_{s_k \in S} w_{s_k}|s_k| \\
S &= \{s_{sub}, s_{sup}, s_{sib}, s_{rel}\} \\
w &= \{1, 0.25, 0.5, 0.5\}
\end{aligned} \tag{5.11}
$$

5.2.7 Semantic Similarity Measure

The Semantic Similarity Measure (SSM) calculates how close the concepts of interest are laid out in the ontology structure. The idea is, if the concepts are positioned relatively far from each other, then it becomes unlikely for those concepts to be represented in a compact manner, rendering their extraction and reuse more difficult. $score_{_{SSM}}(O,Q)$ is the semantic similarity measure score of ontology O for a given query Q. It is a collective measure of the shortest path lengths for all classes that match the query string.

$$
\begin{aligned}
score_{_{SSM}}(O,Q) &= \frac{1}{z}\sum_{i=1}^{z-1}\sum_{j=i+1}^{z}\Psi(r_i,r_j) : \forall_{q\in Q}((r_i,r_j) \in \sigma_O)) \\
\Psi(r_i,r_j) &= \begin{cases} \frac{1}{length(min_{p\in P}\{r_i \xrightarrow{p} r_j\})} & \text{if i} \neq \text{j} \\ 1 & \text{if i} = \text{j} \end{cases} \\
z &= |(r_i,r_j)|
\end{aligned} \tag{5.12}
$$

5.2.8 Betweenness Measure

The Betweenness Measure (BM) [Alani et al., 2006] is a measure for a class on how many times it occurs on the shortest path between other classes. This measure is rooted in the assumption that if a class has a high betweenness value in an ontology, then this class is graphically central to that ontology. The betweenness value of an ontology is the function of the betweenness value of each queried class in the given ontologies. The ontologies where those classes are more central receive a higher BM value.

$score_{_{BM}}(O,Q)$ is the average betweenness value for ontology O and k is the number of matched resources from O for Q. The betweenness measure for resource r_j i.e. $\vartheta(r_j,q_i)$ is computed as:

$$
\begin{aligned}
score_{_{BM}}(O,Q) &= \frac{1}{k}\sum_{i=1}^{n}\sum_{j=1}^{m}\vartheta(r_j,q_i) : r_j \in \sigma_O(q_i) \\
\vartheta(r_j,q_i) &= \sum_{r_x \neq r_y \neq r_j} \frac{\lambda(r_x,r_y(r_j))}{\lambda(r_x,r_y)}
\end{aligned} \tag{5.13}
$$

where $\lambda(r_x, r_y)$ is the number of the shortest path from r_x and r_y and $\lambda(r_x, r_y(r_j))$ is the number of shortest paths from r_x and r_y that passes through r_j.

5.3 Experiment setup

To compare and evaluate the implemented ranking models we developed a benchmark suite named CBRBench, for Canberra Ontology Ranking Benchmark, which includes a collection of ontologies, a set of benchmark queries and a ground truth established by human experts. The CBRBench suite is available at https://zenodo.org/record/11121.

5.3.1 Benchmark Ontology collection

To the best of our knowledge, there exists no benchmark ontology collection for ranking of ontologies. To derive at a representative set of ontologies used on the Web, we used the namespaces registered at prefix.cc[1] as our set of seed ontology URIs. We crawled all registered prefix URIs and for each successfully retrieved ontology (we encountered hundreds of dead links and non-ontology namespaces) we also followed its import statements until no new ontologies were found. This resulted in 1022 ontologies that we used as our benchmark collection. In total, these ontologies define more than 5.5M triples, including ~280k class definitions and ~7.5k property definitions. We stored each ontology separately as a named graph in a Virtuoso database.

5.3.2 Benchmark query terms

To test the ranking algorithms on a representative set of query terms we have used the query log[2] of the Linked Open Vocabularies (LOV) search engine [Vandenbussche and Vatant, 2014] as input. We ranked the most popular search terms in the log covering the period between 06/01/2012 and 16/04/2014 based on their popularity. For the most popular query terms, we checked through a boolean search if there is a representative sample of relevant resources available in our benchmark ontology collection that at least partially match the query term. We included ten search terms in our corpus where there were at least ten relevant ontology classes in the result set. The chosen search terms and their popularity rank within the Linked Open

[1] http://www.prefix.cc
[2] See http://lov.okfn.org/dataset/lov/stats/searchLog.csv

Table 5.2: Query terms

Search Term	Rank
person	1
name	2
event	3
title	5
location	7
address	8
music	10
organization	15
author	16
time	17

Table 5.3: Ranking of *'Person'* in ground truth

URI	Rank
http://xmlns.com/foaf/0.1/Person	1
http://data.press.net/ontology/stuff/Person	2
http://schema.org/Person	3
http://www.w3.org/ns/person#Person	4
http://www.ontotext.com/proton/protontop#Person	5
http://omv.ontoware.org/2005/05/ontology#Person	6
http://bibframe.org/vocab/Person	7
http://iflastandards.info/ns/fr/frbr/frbrer/C1005	8
http://models.okkam.org/ENS-core-vocabulary.owl#person	9
http://swat.cse.lehigh.edu/onto/univ-bench.owl#Person	9

Vocabularies search log are shown in Table 5.2. All queries are single word queries – that is for two reasons. First, only about 11% of all queries posed on the LOV search engine use compound search queries and no compound query was among the 200 most used queries and second, for no compound query in the top 1,000 query terms did the benchmark collection contain enough relevant resources to derive at a meaningful ranking.

Although shallow evaluation schemes are preferred in web search engine evaluations [Pound et al., 2010] we opted for a deep evaluation scheme for two reasons. First, there is only a limited set of knowledge domains where there is a sufficient number of ontologies available on the Web, and second, for the domains with an adequate number of ontologies, many ontologies exist that define or refine similar concepts. This assumption is confirmed by the high number of matching classes for the terms in our query set (see for example Table 5.3).

5.3.3 Establishing the ground truth

We conducted a user study with ten human experts who were sourced from the Australian National University, Monash University, the University of Queensland and the CSIRO. Eight of the evaluators considered themselves to possess *'Expert knowledge'* and two considered themselves to have *'Strong knowledge'* in ontology engineering on a 5-point Likert-Scale from *'Expert knowledge'* to *'No Knowledge'*. All of the evaluators have developed ontologies before, and some are authors of widely cited ontologies. To reduce the number of classes, our ten judges had to score for a given query term (for some query terms a naïve string search returns more than 400 results) we asked two experts to pre-select relevant URIs. The experts were asked to go through all resources that matched a query through a naïve string search and evaluate if the URI is either *'Relevant'* or *'Irrelevant'* for the given query term. We asked the two experts to judge URIs as *'Relevant'* even when they are only vaguely related to the query term, i.e. increasing the false positive ratio.

We developed an evaluation tool which allowed our experts to pose a keyword query for the given term that retrieves all matching ontology classes in the search space. Since keyword queries where the intended meaning of the query is unknown are still the prevalent form of input in Semantic Search [Pound et al., 2010] and since the meaning of the search terms derived from our real query log was also unknown, we needed to establish the primary intention for each of our query terms. We used the main definition from the Oxford dictionary for each term and included it in the questionnaire for our judges. We then asked our ten human experts to rate the relevance of the results to each of the 10 query terms from Table 5.2 according to their relevance to the definition of the term from the Oxford dictionary. After submitting the keyword query, each evaluator was presented with a randomly ordered list of the matching ontology classes in the search space to eliminate any bias. For each result we showed the evaluator, the `URI`, the `rdfs:label` and the `rdfs:comment`, the properties of the class and its super-classes and sub-classes. A judge could then rate the relevance of the class with radio buttons below each search result on a 5-point Likert scale with values *'Extremely Useful'*, *'Useful'*, *'Relevant'*, *'Slightly Relevant'* and *'Irrelevant'*.

There was no time restriction for the judges to finish the experiment. We assigned values from 0-4 for *'Irrelevant'-'Extremely Useful'* for each score and performed a hypothesis test on the average scores per evaluator with a H_0 $\mu = 2$ against H_1 $\mu <> 0$. This resulted in a p-value of 0.0004, a standard error of the mean of 0.144 and a 95% confidence interval for the mean score of (0.83,1.49), indicating there is strong evidence that the average scores per evaluator are not two which would indicate a randomness of the scores. We also asked our ten evaluators to score 62 random response URIs for the ten queries again two months after we performed our initial experiment. The average scores of the ten evaluators for these URIs had a correlation coefficient of 0.93, indicating that in average, the scores of the participants in the second study were highly correlated with the scores in the first study.

Table 5.3 shows the ideal ranking for the query *'Person'* as derived from the me-

dian relevance scores from our ten experts. For ties, we considered the resource with the more consistent relevance scores (i.e., the lower standard deviation) as better ranked. Not all ties could be resolved in this way as can be seen in Table 5.3 for rank No. 9.

5.3.4 Evaluation and Performance Measures

We consider three popular metrics from the information retrieval community, precision at k (P@k), mean average precision (MAP), and normalised discounted cumulative gain (NDCG). Since we asked our judges to assign a non-binary value of relevance (on a 5-point Likert scale), we converted these values to a binary value for all those metrics that consider a binary notion of relevance. We chose a resource as being relevant to the query term if the relevance score is equal or higher than the average value on the 5-point Likert scale. Changing this cut-off value to the right or the left of the average changes the overall precision of the result. However, the relative performance of the algorithms remains the same.

Precision@k: We are calculating precision at k (P@k) for a k value of 10. P@k in our experiment is calculated as:

$$p@k \quad = \quad \frac{number\ of\ relevant\ documents\ in\ top\ k\ results}{k}$$

Average Precision: The average precision for the query Q of a ranking model is defined as:

$$AP(Q) \quad = \quad \frac{\sum_{i=1}^{k} rel(r_i) * P@i}{k}$$

where $rel(r_i)$ is 1 if r_i is a relevant resource for the query Q and 0 otherwise, $P@i$ is the precision at i and k is the cut off value (i.e. 10 in our experiment). MAP is defined as the mean of AP over all queries run in this experiment and is calculated as:

$$MAP \quad = \quad \frac{\sum_{Q \in \mathbb{Q}} AP(Q)}{|\mathbb{Q}|}$$

Normalize Discounted Cumulative Gain (NDCG): NDCG is a standard evaluation measure for ranking tasks with non-binary relevance judgement. NDCG is defined based on a gain vector G, that is, a vector containing the relevance judgements at each rank. Then, the discounted cumulative gain measures the overall gain obtained by reaching rank k, putting more weight at the top of the ranking:

$$DCG(Q) = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(1+i)}$$

To compute the final NDCG, we divide DCG by its optimal value iDCG which puts the most relevant results first. iDCG is calculated by computing the optimal gain vector for an ideal ordering obtained from the median of the user assigned relevance scores.

5.4 Results

Table 5.4, 5.5, 5.6 and 5.7 show the Precision, the AP, the DCG and the NDCG scores respectively, for all ranking models for each query term, whereas average of these metrics on all ten queries is shown in Table 5.8. Figure 5.1 shows the P@10, AP, DCG, NDCG scores for each of the eight ranking models on all ten queries. For P@10 and AP, tf-idf is the best performing algorithm with betweenness measure as the second best and PageRank as the third best. Regarding the correct order of top k results, we found tf-idf again as the best performing algorithm, with betweenness measure and PageRank as the second and third best, respectively.

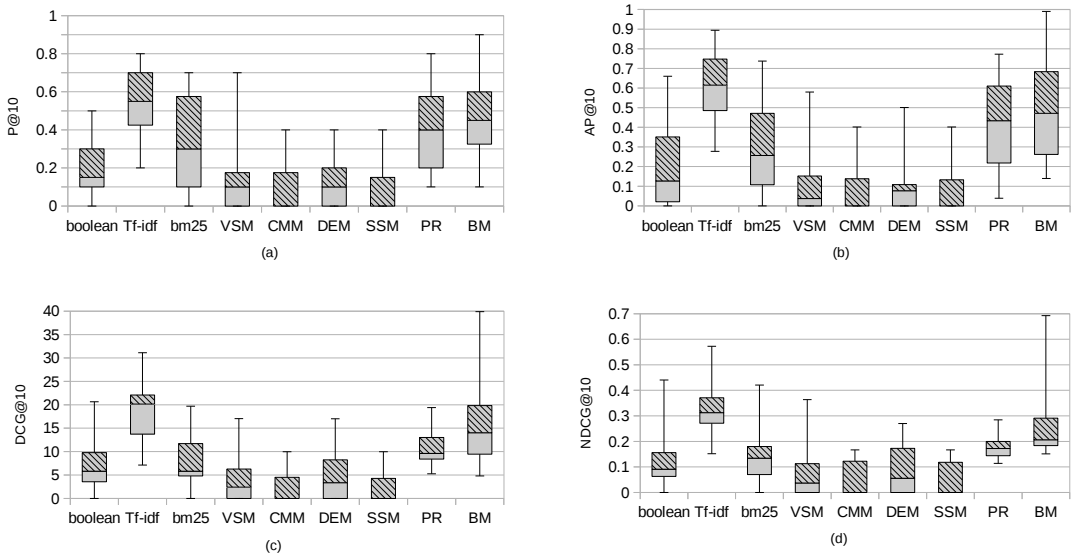


Figure 5.1: Effectiveness of Ranking Model

Table 5.4: Precision @ 10

	Person	Name	Event	Title	Location	Address	Music	Organization	Author	Time
boolean	0.30	0.00	0.40	0.10	0.00	0.50	0.30	0.10	0.10	0.20
tf-idf	**0.70**	0.40	**0.70**	0.50	**0.60**	0.80	**0.40**	**0.80**	0.20	**0.50**
BM25	0.10	**0.70**	0.10	0.40	0.10	0.50	0.20	0.60	**0.60**	0.00
vector-space	0.10	0.00	0.10	0.30	0.00	0.70	0.20	0.00	0.10	0.00
pagerank	0.10	0.20	0.70	**0.60**	0.40	0.50	0.20	0.10	0.40	0.30
class-match-measure	0.00	0.00	0.00	0.40	0.00	0.30	0.20	0.00	0.10	0.00
density-measure	0.20	0.00	0.10	0.10	0.00	0.40	0.20	0.00	0.20	0.00
semantic-similarity	0.00	0.00	0.00	0.40	0.00	0.30	0.20	0.00	0.00	0.00
between-measure	0.60	0.30	0.60	0.40	**0.60**	**0.90**	0.10	0.70	0.40	0.40

Table 5.5: AP @ 10

	Person	Name	Event	Title	Location	Address	Music	Organization	Author	Time
boolean	0.17	0.00	0.23	0.02	0.00	0.66	0.39	0.02	0.08	0.44
tf-idf	**0.75**	0.44	**0.82**	0.51	**0.73**	0.89	**0.48**	0.70	0.28	0.53
BM25	0.19	**0.74**	0.03	0.40	0.08	0.49	0.18	0.32	**0.62**	0.00
vector-space	0.06	0.00	0.19	0.08	0.00	0.58	0.18	0.00	0.01	0.00
pagerank	0.19	0.38	0.55	**0.70**	0.63	0.18	0.04	0.29	0.49	**0.77**
class-match-measure	0.00	0.00	0.00	0.40	0.00	0.35	0.18	0.00	0.02	0.00
density-measure	0.30	0.00	0.08	0.11	0.00	0.50	0.11	0.00	0.07	0.00
semantic-similarity	0.00	0.00	0.00	0.40	0.00	0.35	0.18	0.00	0.00	0.00
between-measure	0.69	0.23	0.40	0.36	0.55	**0.99**	0.14	**0.80**	0.14	0.66

Table 5.6: DCG @ 10

	Person	Name	Event	Title	Location	Address	Music	Organization	Author	Time
boolean	4.36	0.58	10.54	8.45	0.00	17.76	8.36	4.19	3.98	7.87
tf-idf	**15.91**	8.00	**25.29**	15.21	18.51	20.91	**12.52**	21.56	5.32	**13.99**
BM25	5.44	**16.88**	2.16	10.47	8.43	14.30	9.28	11.96	**8.14**	0.41
vector-space	6.98	2.32	4.94	9.47	0.08	14.92	9.28	0.09	2.29	0.41
pagerank	14.08	1.03	11.28	**17.55**	2.48	22.68	5.77	22.24	5.49	9.38
class-match-measure	0.65	1.60	0.34	11.60	0.00	8.74	9.28	0.00	4.61	0.41
density-measure	12.85	0.07	4.99	7.84	0.04	10.82	6.62	0.00	2.61	0.00
semantic-similarity	0.65	1.60	0.34	11.60	0.00	8.74	9.28	0.00	3.82	0.41
between-measure	12.26	7.48	11.73	17.06	**21.06**	**25.59**	6.37	**30.16**	7.04	12.72

Table 5.7: NDCG @ 10

	Person	Name	Event	Title	Location	Address	Music	Organization	Author	Time
boolean	0.06	0.00	0.16	0.11	0.00	0.44	0.22	0.07	0.07	0.15
tf-idf	**0.29**	0.20	**0.46**	**0.27**	**0.32**	0.57	**0.39**	0.32	0.15	**0.30**
BM25	0.07	**0.42**	0.02	0.13	0.07	0.32	0.16	0.19	0.14	0.00
vector-space	0.12	0.00	0.06	0.10	0.00	0.36	0.16	0.00	0.01	0.00
pagerank	0.14	0.18	0.28	0.21	0.15	0.18	0.17	0.22	0.11	0.14
class-match-measure	0.00	0.00	0.00	0.15	0.00	0.17	0.16	0.00	0.05	0.00
density-measure	0.25	0.00	0.07	0.13	0.00	0.27	0.19	0.00	0.04	0.00
semantic-similarity	0.00	0.00	0.00	0.15	0.00	0.17	0.16	0.00	0.03	0.00
between-measure	0.22	0.18	0.17	0.24	0.31	**0.69**	0.15	**0.59**	**0.19**	0.19

Table 5.8: Overall average scores

	P@10	MAP	DCG	NDCG
boolean	0.20	0.201	6.609	0.128
tf-idf	**0.56**	**0.613**	15.722	**0.327**
BM25	0.33	0.306	7.940	0.152
vector-space	0.15	0.110	4.177	0.082
class-match-measure	0.10	0.095	2.540	0.053
pagerank	0.40	0.423	10.497	0.177
density-measure	0.12	0.117	5.128	0.095
semantic-similarity	0.09	0.093	2.456	0.051
between-measure	0.48	0.496	**16.975**	0.29

5.4.1 Results Analysis

From the results of this experiment it can be seen that, somehow surprisingly, content-based models (i.e. tf-idf and BM25) outperform the graph-based ranking models for most queries. Overall, seven out of ten times, the content-based models achieve better or equal results compared to the highest NDCG scores for all ranking algorithms.

However, although tf-idf achieved the highest mean average precision value of 0.6 in our experiment, it is still far from an ideal ranking performance. This is because the philosophy of tf-idf works well for the tf part, but not so for the idf part when ranking resources in ontologies. The intuition behind tf-idf is that if a word frequently appears in a document, it is important to the document and is given a high score (i.e. tf value), but if it appears in many documents, it is not a unique identifier and is given a low score (i.e. idf value). In ontologies, a resource that is reused across many ontologies is a popular and relatively more valuable resource in the ontology and the corpus. Therefore, in our experiment, tf-idf successfully ranks a resource high in the result set if that resource is the central concept of the ontology (i.e. it is assigned a high tf value). However, if a resource is also popular among the corpus, it is scored down for the idf value. For example, $\mathrm{http://xmlns.com/foaf/0.1/Person}$ has the highest tf value (i.e. 0.589) of all concepts in the FOAF ontology, but since it is also the most popular concept in our corpus appearing in total in 162 distinct ontologies, it does not appear among the top ten results of tf-idf.

Since BM25 is a cumulative relevance score for an ontology rooted in the tf and idf values of a matched resource, the limitations of tf-idf are implicit in BM25 as well. However, BM25 ranks concept specific ontologies higher in the result set for a query term that matches to that particular concept. The reason is that for a specific ontology, the query term matches to one of the important resource and many of its attached resources. All these matched resources sum up to a higher BM25 score for that ontology. For example, for the *'Name'* query, BM25 ranks all resources in

the GND ontology[3] higher since this ontology defines different types of Names. All these types of names are important concepts of this ontology and finally leverage the BM25 score for the GND ontology.

The vector space model did not perform well for any query. The main reason is that the vector space model considers tf-idf values of resources as well as query term/s. The idf value for a query term is calculated by considering the idf values of all the resources in the corpus that matched the query term. Therefore, the effect of the wrong assumptions for the idf values doubles for the vector space model. PageRank ranks resources according to their popularity; that is why it performs, for example, well in ranking highly the *'Person'* concept in the FOAF ontology as it is a widely used ontology that is imported by many other ontologies. However, considering popularity in the corpus as the only factor for ranking ontologies sometimes results in poor precision and recall. e.g. http://www.loria.fr/~coulet/ontology/sopharm/version2.0/disease_ontology.owl#DOID_4977 with the label *'other road accidents injuring unspecified person'* is one of the popular resources in our corpus but not at all relevant for the *'Person'* concept. Still, PageRank assigns it a higher rank based on its popularity in the corpus. The performance of the PageRank algorithm could be significantly improved if it also takes the data for a given ontology into consideration (as is done in Semantic Search engines). Instead of using only the import statement as the measure of popularity, the links from data will give higher weights to resources in ontologies for which there exists data across multiple domains.

As expected, the class match measure is the least precise algorithm in the experiment. Since the algorithm ranks an ontology only on the basis of the label of the matched resources within that ontology, an ontology with single or zero exact matched labels and many partial match labels gets a higher relevance score than those ontologies where few concepts are relatively more important. Secondly, assigning the same weight to partially matched labels is problematic. For example, for the query *'Address'* two partially matched resources *'Postal address'*[4] and *'Email address of specimen provider principal investigator'*[5] are obviously not equally relevant to the address definition provided in our user study. However, CMM uses equal weights for both of these resources while computing the relevance score of their corresponding ontologies.

The density measure model performs relatively poorly because it assigns high weights for superclass and subclass relations. The intention is that the further specified a resource is in an ontology the more important it is. However, in our study the density measure model always favours upper-level ontologies or highly layered ontologies, where many subclasses and superclasses are defined for a resource (e.g. OBO ontologies), irrespective of its relevance to the query term.

The semantic similarity measure model considers the proximity of matched resources in an ontology. Although this metric can be useful when considering similarity among the matched resources of two or more query terms of a multi-keyword

[3] http://d-nb.info/standards/elementset/gnd#
[4] http://purl.obolibrary.org/obo/IAO_0000422
[5] http://purl.obolibrary.org/obo/OBI_0001903

query, it performs poorly on single word queries. As mentioned earlier, users seem to be not using multi-keyword queries in ontology search yet, and thus the semantic similarity measure appears to be not particularly useful for ranking resources in ontologies.

The betweenness measure performs better than all other graph-based ranking models because it calculates the relative importance of the resource to the particular ontology. A resource with a high betweenness value is the central resource of that ontology [Alani et al., 2006], which means that the resource is well defined and important to the ontology. Further, the betweenness measure performs well even with resources that are irrelevant to the query term if they are not central resources of that ontology, as their score will not contribute significantly to the cumulative relevance score for the ontology. For example, irrelevant resources such as *'dislocation'* for the query *'location'* do not appear high in the ranking of the betweenness measure, because all resources with the label including *'dislocation'* are not central concepts in the ontology where they are defined.

A general observation that can be made is that all ranking models other than tf-idf ignore the relevance and importance of a resource to the query when assigning a weight to a particular ontology for a given query term. This is more prominent for graph-based approaches, where the cumulative ranking score for an ontology is computed based on all the relevant terms of that ontology for this query. An ontology that has more matched URIs to the query term gets a higher weight than an ontology that has few or only a single relevant resource in the ontology. For example, http://www.ontologydesignpatterns.org/cp/owl/participation.owl#Event with the label *'event'* is ranked *'Extremely useful'* to *'Useful'* for the query *'event'* by our human experts. However, since this is the only relevant resource in the ontology and it is a small ontology, none of the graph-based models ranked this URI among the top ten resources.

5.5 Recommendations

Based on the analysis of our experiment we put forward the following four recommendations that we believe could significantly improve the performance of the different ranking algorithms.

Intended type vs. context resource: We believe that differentiating between the intended type and the context resource of a URI has a positive impact on the performance of all ranking models. For example, for a resource in the GND ontology[6] with the label *'Name of the Person'*, *'Name'* is the intended type, whereas *'Person'* is the context resource. This resource URI appears in the search results for both, the *'Person'* and the *'Name'* query term in our experiment. The human experts ranked this resource on average from *'Extremely useful'* to *'Useful'* for the *'Name'* query term and only *'Slightly useful'* for the *'Person'* query. However, all the ranking algorithms assigned an equal weight to this resource while

[6] http://d-nb.info/standards/elementset/gnd#NameOfThePerson

calculating ranks for either of the two query terms. The performance of the ranking models could be improved if they either only consider those resource URIs whose intended type matches the query's intended type or if they assign a higher weight to such URIs as compared to the ones where the query terms' intended type matches only the context resource of that URI.

Exact vs. partial matches: As identified by [Alani et al., 2006] exact matching should be favoured over partial matching in ranking ontologies. Whereas the class match measure model assigns a value of 0.6 to exact matches and 0.4 to partial matches. For example, for the query 'Location', results that include *'dislocation'* as partial matches should not be considered, since the word sense for location and dislocation are different. Instead of assigning static weight factors, we believe that other means of disambiguation between the actual meaning of the query term and the resource URI can significantly improve the performance of the algorithms. Wordnet [Miller, 1995] or a disambiguation at the time of entry of the query term could be efficient methods for this purpose.

Relevant relations vs. context relations: For the graph-based ranking models that calculate the relevance score according to the number of relationships for the resource within that ontology (i.e. density measure and betweenness measure), direct properties, subclasses and superclasses of a class have to be distinguished from relations (i.e. properties) that are very generic and are inferred from its super-classes. For example, the class *'email address'*[7] from one of the OBO ontologies has properties like *'part of continuant at some time'*, *'geographic focus'*, *'is about'*, *'has subject area'*, *'concretized by at some time'*, *'date/time value'* and *'keywords'*. However, not all of these properties are relevant to the concept *'email address'*.

Resource relevance vs. ontology relevance: All ranking models discussed in this study (except tf-idf), rank ontologies for the query term by considering all matched resources from a given ontology against the query term. This results in a global rank for the ontology and all the resources that belong to that ontology share the same ontology relevance score. Therefore, in a result set, many resources hold the same relevance score. While ordering resources with the same relevance score from the ontology, the ranking models lack a mechanism to rank resources within the same ontology. We believe that the tf value of the resource could be a good measure to assign scores to resources within an ontology. Therefore, while ranking all the resources of an ontology, the tf value can be used to further rank resources that belong to the same ontology. Another solution could be to compute individual measures (all measures other than tf-idf) for each resource, independent of how many other matched resources there are in the same ontology.

We test most of these recommendations by implementing them in Chapter 6 and 8. The intended type of a resource is implemented as a filter on matched results

[7] http://purl.obolibrary.org/obo/IAO_0000429

in Section 6.4.3, and the evaluation results presented in Section 6.5.2.2 show that it enhances the overall effectiveness of the concept retrieval framework. In Section 6.3, the proposed ranking model computes the relevance of a resource contrary to the ontology relevance, and Section 6.5.2.1 presents that the proposed algorithm is more effective in ontology retrieval than the algorithms evaluated in this chapter. Moreover, exact and partial matches contribute differently towards the overall relevance score of an ontology for a user query in Section 8.3.2.1. The results presented in Section 8.4 show an improved effectiveness as compared to the state of the art ranking models.

5.6 Summary

This chapter represents, to the best of our knowledge, the first systematic attempt at establishing a benchmark for ontology ranking. We established a ground truth through a user study with ten ontology engineers that we then used to compare eight state-of-the-art ranking models. When comparing the ranking models to the ideal ranking obtained through the user study, we observed that content-based ranking models (i.e. tf-idf and BM25) slightly outperform graph-based models in aspects such as betweenness measure. Even though content-based models performed best in this study, the performance is still inferior to the performance of the same models on ranking documents because of the structural differences between documents and ontologies. We put forward four recommendations that we believe can considerably improve the performance of the discussed models for ranking resources in ontologies. In particular:

- ***Determine the intended type of a resource:*** A resource should only match a query if the intended type of the query matches the intended type of the resource.
- ***Treat partial matches differently:*** Instead of treating partial matches of the query and a resource similar to exact matches or assigning a static weight factor, the models should consider other means of disambiguating the actual meaning of the query when matching it with a resource.
- ***Assign the higher weight to direct properties:*** Instead of considering all relations for a class equally when calculating the centrality score in graph-based models, the models should consider assigning a higher score to relations that describe the class directly.
- ***Compute a resource relevance:*** In addition to computing a relevance score for an ontology as a whole, all ranking models should be changed so that they also compute a score for individual resources within the ontology.

We use the CBRBench presented in this Chapter for the evaluation of ranking models presented in Chapter 6 and 7. Moreover, we leverage CBRBench queries to evaluate the work presented in Chapter 8.

Chapter 6

Relationship-based Concept Retrieval

In this chapter, we introduce a relationship-based concept retrieval framework to address the research question **RQ5** i.e., what are the inherent characteristics of ontologies that can be used to rank relevant resources and ontologies for keyword queries? As part of this framework, DWRank a two-staged bi-directional graph-walk ranking algorithm is proposed to rank concepts in ontologies based on how well they represent a given search term. DWRank is an effort towards addressing the research question **RQ3** i.e., how to rank relevant resources and ontologies for keyword queries? We apply this algorithm to the task of searching and ranking concepts in ontologies and compare it with state-of-the-art ontology ranking models and traditional information retrieval algorithms such as PageRank and tf-idf.

6.1 Introduction

As discussed earlier, the widespread use of ontologies due to growth in Linked data necessitates the search engines or libraries to discover existing ontologies and the concepts and relations within. There are several established ontology libraries in vertical domains such as the Open Biological and Biomedical Ontologies library[1] or the BioPortal [Noy et al., 2009], where keyword queries are still the preferred method to find concepts and relations in the registered ontologies. However, since there may exist many ontologies that contain concepts and relations with their label matching the keyword query, the matches need to be usefully ranked. There has been some previous work, for example [Ding et al., 2005; Alani et al., 2006; Noy et al., 2009; Noy and d'Aquin, 2012], to tackle the problem of finding and ranking ontologies. More recently, also dedicated ontology search engines have emerged [Vandenbussche and Vatant, 2014], but the ranking algorithms they use mostly do not contemplate the inherent structure and semantics of ontologies [Butt et al., 2014a]. Therefore, a ranking model that leverage the ranking quality of search results through the use of the ontology structure is paramount to satisfy the information need of ontology search users.

[1] http://www.obofoundry.org/

In this chapter, we propose a new ontology concept retrieval framework that uses a number of techniques to rate and rank each concept in an ontology based on how well it represents a given search term. The ranking in the framework is conducted in two phases. First, our offline ranking algorithm, DWRank, computes the centrality of a concept within an ontology based on its connectivity to other concepts within the ontology itself. Then, the authority of a concept is computed which depends on the number of relationships between ontologies and the weight of these relationships based on the authority of the source ontology. The assumption behind this is that ontologies that reuse and are reused by other ontologies are more authoritative than others. In a second, online query processing phase a candidate set for a *top-k* concept is selected from the offline ranked list of ontologies and then filtered based on two strategies, the diverse results semantics and the intended type semantics. The resulting list of *top-k* of concepts is then evaluated against a ground truth derived through a human evaluation published previously [Butt et al., 2014a]. Our evaluation shows that the proposed framework significantly outperforms the state-of-the-art ranking models on the task of finding concepts in ontologies for all ten benchmark queries in the ontology collection.

6.2 Concept Retrieval Framework

In the following, we first define the terms used throughout the chapter. We then give a brief overview of the mechanics of the ranking framework.

6.2.1 Preliminaries

An ontology here refers to a graph based formalisation $O = (V, E, L)$ of a domain knowledge. V is a finite set of nodes where $v \in V$ denotes a domain concept in O, E is the edge set where $(v, v') \in E$ denotes an explicit or implicit relationship between v and v'. L is a labelling function which assigns a label $L(v)$ (resp. $L(e)$ or $L(O)$) to node v (resp. an edge $e \in E$ or the ontology O). In practice the labelling function L may specify (1) the node labels to relate the node to the referent concept, e.g. person, place and role; and (2) the edge labels as explicit relationships between concepts e.g., friendship, work and participation or implicit relationships e.g., sub-concept and super-concept, and (3) the ontology label to relate the ontology to the domain or some identity.

6.2.1.1 Intra-Ontology Relationships

An *intra-ontology relationship* $I_a = ((v, v'), O)$ is a directed edge (v, v'), where $(v, v') \in E(O)$ for $v \in V(O)$ and $v' \in V(O)$.

6.2.1.2 Inter-Ontology Relationships

An *inter-ontology relationship* I_e = *((v,v′), O, O′)* is a directed edge (*O*, *O′*), where (*v*, *v′*) ∈ E(O), *L(v) = L(O)* , *L(v′) = L(O′)* and *L (v,v′) = owl:imports*[2].

6.2.1.3 Forward Link Concepts

Forward link concepts $C_{FLinks}(v, O)$ is a set of concepts V' in an ontology O, where $V' \subset V(O)$ and $\forall\ v_i \in V'$, $\exists\ (v, v_i) \in E(O)$.

6.2.1.4 Back Link Concepts

Back link concepts $C_{BLinks}(v, O)$ is a set of concepts V'' in an ontology O, where $V'' \subset V(O)$ and $\forall\ v_j \in V''$, $\exists\ (v_j, v) \in E(O)$.

6.2.2 Overview of the framework

The framework is composed of two phases as shown in Figure 6.1. The first phase is an offline phase where two indices, i.e. *ConHubIdx* and *OntAuthIdx*, are constructed for the whole ontology corpus. The second phase is an online query processing phase where a query is evaluated, and the *top-k* concepts are returned to the user.

6.2.2.1 Offline Ranking and Index construction

The framework first constructs a *ConHubIdx* on all concepts and an *OntAuthIdx* on all ontologies in the ontology corpus ○. The *ConHubIdx* maps each concept of an ontology to its corresponding *hub score*. Similarly, the *OntAuthIdx* maps each ontology to its precomputed *authority score*. The *hub score* and *authority score* are defined in Section 6.3.1

6.2.2.2 Online Query Processing

Upon receiving a query ℚ, the framework extracts the *candidate result set* $C_Q = \{(v_1, O_1), ..., (v_i, O_j)\}$ including all matches that are semantically similar to ℚ by querying the ontology repository. The *hub score* and *authority score* for all $(v, O) \in C_Q$ are extracted from the corresponding indices as $H(C_Q)$ and $A(C_Q)$ lists. A ranked list $R(C_Q)$ of a candidate result set is computed from $H(C_Q)$ and $A(C_Q)$ along with the text relevancy measure. $R(C_Q)$ is further filtered to satisfy two result set properties, i.e. the *Diverse Result Semantics* and the *Intended Type Semantics*, as introduced in Section 6.4.3.

[2] http://www.w3.org/2002/07/owl#imports

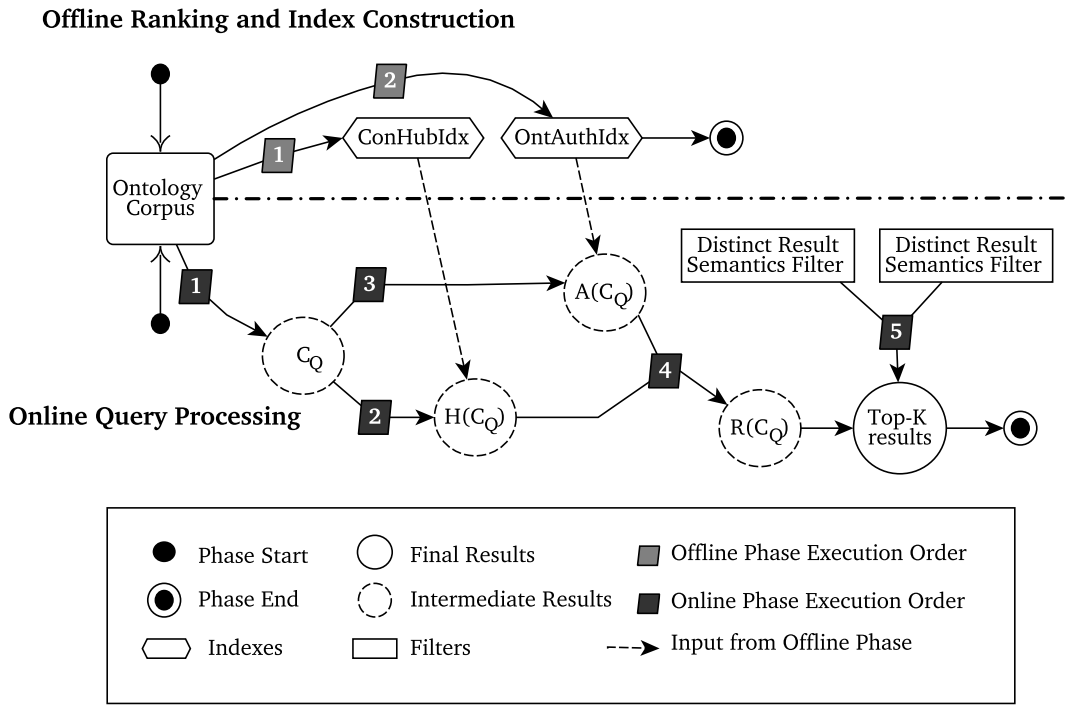


Figure 6.1: Relationship-based top-k concept retrieval framework

6.3 Offline Ranking and Index Construction

In this section the offline ranking phase of the *relationship-based top-k concept retrieval* framework is described (cf. Figure 6.2). First, we introduce the ranking model in Section 6.3.1 and then we introduce the index construction based on the ranking model in Section 6.3.2.

6.3.1 DWRank: A Dual Walk based Ranking Model

Our ranking model characterises two features of a concept to determine its rank in a corpus:

1. A concept is more important, if it is a *central concept* to the ontology within which it is defined.

2. A concept is more important, if it is defined in an *authoritative* ontology.

More precisely, first, the offline ranking module generates for each concept in the corpus a *hub score*, a measure of the *centrality of a concept*, i.e. the extent that the concept is `related` to the domain for which the ontology is formalised. Second, the *authority score* is generated as a measure of the `authoritativeness` of the ontology. A link analysis algorithm, i.e. PageRank, is performed that leverages the ontological

structure and semantics to compute these scores. However, the difference between our model and traditional PageRank-like algorithms is two-fold. Firstly, we perform the link analysis independently on each ontology to find a *hub score* and then only on the whole ontology corpus considering an ontology as a node and *inter-ontology relationships* as links. Secondly, we differentiate the type of relationship (i.e. inter-ontology and intra-ontology), and the direction of the walk varies by the kind of the relationship.

Our Model *DualWalkRank* is named after its characteristic of a dual directional walk to compute the ranks of concepts.

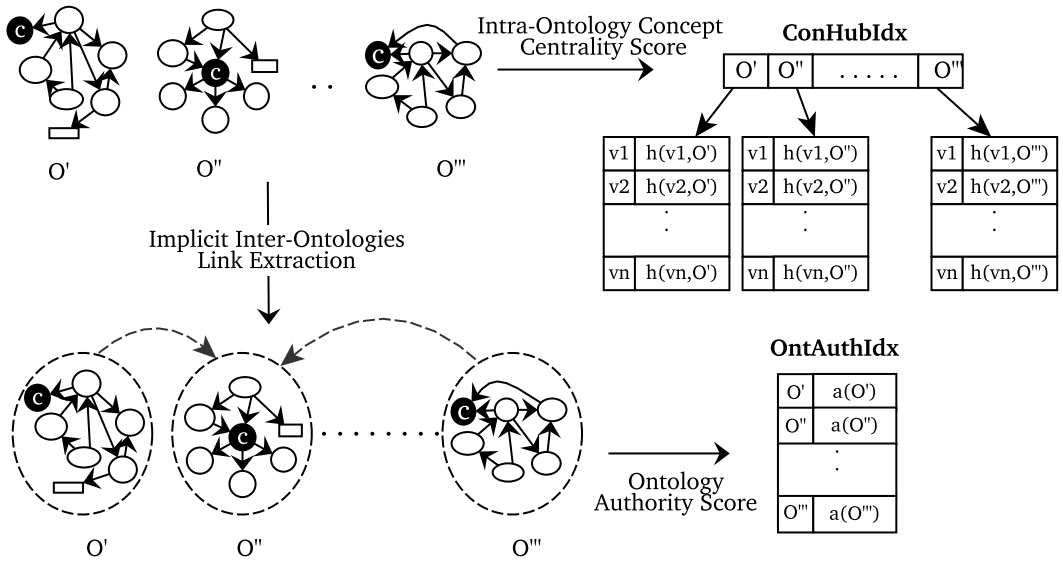


Figure 6.2: Offline Index Construction

6.3.1.1 HubScore: The centrality of a concept within an ontology

The *hub score* is a measure of the centrality of a concept within an ontology. We define a *hub function* `h(v,O)` that calculates the *hub score*. The *hub function* is characterised by two features:

- **Connectivity:** A concept is more central to an ontology, if there are more *intra-ontology relationships* starting from the concept.
- **Neighbourhood:** A concept is more central to an ontology, if there is an *intra-ontology relationship* starting from the concept to another central concept.

According to these features, a concept accepts the centrality of another concept based on its forward link concepts (like a `hub`). The *hub function* is, therefore, a complete reverse of the PageRank algorithm [Page et al., 1998] where a node accepts scores from its referent nodes i.e. backlink concepts.

We adopt a Reverse-PageRank [Fogaras, 2003] as the *hub function* to find the centrality of a concept within the ontology. The hub function is an iterative function and at any iteration k, the *hub function* is featured as deinfed in Equation 6.1.

$$h_k(v,O) = \sum_{v_i \in C_{FLinks}(v,O)} \frac{h_{k-1}(v_i,O)}{|C_{BLinks}(v_i,O)|} \tag{6.1}$$

Within the original PageRank framework, there are two types of links in a graph, strong and weak links. The links that exist in the graph are *strong links*. *Weak links* are artificially created links by a damping factor α, and they connect all nodes to all other nodes. Since *data-type relationships* of a concept do not connect it to other concepts in an ontology, most PageRank-like algorithms adopted for ontology ranking consider only *object type relationships* of a concept while ignoring others. We adopt the notion of weak links in our *hub function* to be able to also consider *data-type relationships* along with *object-type relationships* for the ontology ranking. We generate a set of artificial concepts *V(O)* in the ontology that act as a sink for every *data-type relationship* and label these concepts with the data type relationship label, i.e. $\forall\ v_j \in V'$, L(v'_j) = L (v_i,v'_j). After incorporating *weak links* and *weak nodes*, Equation 6.2 reflects the complete feature of our *hub function*.

$$h_k(v,O) = \frac{1-\alpha}{|V|} + \alpha * \sum_{v_i \in C_{SFLinks}(v,O) \cup C_{WFLinks}(v,O)} \frac{h_{k-1}(v_i,O)}{|C_{BLinks}(v_i,O)|} \tag{6.2}$$

In Equation 6.2, $C_{SFLinks}(v,O)$ is a set of *strong forward link concepts* and $C_{WFLinks}(v,O)$ is a set of *weak forward link concepts*. Our *hub function* is similar to [Wu et al., 2008a], but varies from it as we consider *weak nodes* and we are not considering relationships weights. The results presented in [Wu et al., 2008a] also justify our choice of ReversePageRank over other algorithms to measure the centrality. We normalise the hub scores of each concept v within an ontology O through the `z-score` of the concept's hub score after the last iteration of the *hub function* as follows:

$$h_n(v,O) \quad = \quad \frac{h(v,O) - \mu_h(O)}{\sigma_h(O)} \tag{6.3}$$

In Equation 6.3, $h_n(v,O)$ is a normalised hub score of v, $\mu_h(O)$ is an average of hub scores of all concepts in the ontology and $\sigma_h(O)$ is the standard deviation of hub scores of the concepts in the ontology.

6.3.1.2 AuthorityScore: The authoritativeness of a concept

The *authority score* is the measure of the authoritativeness of a concept within an ontology. As mentioned earlier, the *authoritativeness of a concept* depends upon the *authoritativeness of the ontology* within which it is defined. Therefore, we define the

authority function `a(O)` to measure the *authority score* of an ontology. Our *authority function* is characterised by the following two features:

- **Reuse:** An ontology is more authoritative, if there are more *inter-ontology relationships* ending at the ontology.
- **Neighbourhood:** An ontology is more authoritative, if there is an *inter-ontology relationship* starting from an authoritative ontology to the ontology.

Based on these two features, an *inter-ontology relationship* $I_e((v, v'), O, O')$ is considered as a "positive vote" for the authoritativeness of ontology O' from O. The PageRank is adopted as the *authority function*, whereby each ontology is considered a node and *inter-ontology relationships* are considered links among nodes. Equation 6.4 formalise the *authority function* which computes the authoritativeness of O at the *kth* iteration.

$$a_k(O) = \frac{1-\alpha}{|O|} + \alpha \sum_{O_i \in O_{BLinks}(O)} \frac{a_{k-1}(O_i)}{|O_{FLinks}(O_i)|} \tag{6.4}$$

In Equation 6.4, $O_{BLinks}(O)$ is a set of *back link ontologies* and $O_{FLinks}(O)$ is a set of *forward link ontologies*. The definition of $O_{FLinks}(O)$ (resp. $O_{BLinks}(O)$) is similar to $C_{FLinks}(v, O)$ (resp. $C_{BLinks}(v, O)$), however, the links are *inter-ontology relationships*.

Similar to the *hub score*, we also compute the `z-score` of each ontology after the last iteration of the *authority function* as follows:

$$a_n(O) \quad = \quad \frac{a(O) - \mu_a(\mathbb{O})}{\sigma_a(\mathbb{O})} \tag{6.5}$$

In Equation 6.5, $a_n(O)$ is the normalised authority score of v, $\mu_a(\mathbb{O})$ is an average of the authority scores of all ontologies in the corpus and $\sigma_a(\mathbb{O})$ is the standard deviation of the authority scores of ontologies in $\mathbb{O}$.

6.3.1.3 DWRank Score

Finally, we define the *DWRank* $R_{(v,O)}$, as a function of the *text relevancy*, the *normalised hub score* and the *normalised authority score*. The function is described as a quantitative metric for the overall relevance between the query $\mathbb{Q}$ and the concept v; and the concept *hub and authority* score as follows:

$$\begin{aligned} R_{(v,O)} &= F_V(v, \mathbb{Q}) * [w_1 h(v, O) + w_2 a(O)] \\ F_V(v, \mathbb{Q}) &= \sum_{q \in Q} f_{ss}(q, \phi(q_v)) \end{aligned} \tag{6.6}$$

In Equation 6.6, w_1 and w_2 are the weights for the *hub function* and the *authority function*. $F_V(v, \mathbb{Q})$ aggregates the contribution of all matched words of a node v, in an ontology O, to the query keywords $q \in \mathbb{Q}$. f_{ss} returns a binary value : it returns 1 if q has a match $\phi(q_v)$ in v, and 0 otherwise. The metric favours the nodes v that are semantically matched to more keywords of the query $\mathbb{Q}$.

6.3.2 Index Construction: An execution of DWRank

In this section, we explain the execution model of *DWRank* and the construction of the indices.

6.3.2.1 ConHubIdx

The ConHubIdx is a bi-level index where each entry in the index maps a concept of an ontology to its *normalised hub score* $h_n(v, O)$ as shown in Figure 6.2 (top left). To construct the *ConHubIdx* for all ontologies in $\mathbb{O}$, (1) the *hub function* is executed in an iterative way to get the *hub score* of all the concepts in ontology O, and (2) after the last iteration, we compute the normalised hub scores and (3) insert the concepts along with their normalised hub scores in an ontology to the index.

6.3.2.2 OntAuthIdx

The OntAuthIdx is an index where each entry in the index maps an ontology to its *normalised authority score* $a_n(O)$ as shown in Figure 6.2 (bottom left). To construct the *OntAuthIdx* on the corpus $\mathbb{O}$, (1) the *authority function* is executed to get an *auth score* of all the ontologies in $\mathbb{O}$, (2) after the last iteration, the normalised authority scores are computed, and (3) the ontology along with its normalised authority scores is inserted as an entry to the index.

6.3.2.3 Inter-Ontology Relationships Extraction

As we mentioned earlier, the *authority function* leverages the *inter-ontology relationships* that are directed links among ontologies. If ontology *OntA* reuses the resources in ontology *OntB*, ontology *OntA* declares the reuse of resources through an OWL import property i.e. `owl:imports`. Since some ontology practitioners fail to declare the reuse of ontologies explicitly, the `owl:imports` relationships in an ontology are often inaccurate representations of the *inter-ontology relationships*.

We, therefore, identify the implicit *inter-ontology relationships* by considering the reused resources in the corpus. Finding the implicit *inter-ontology relationships* involves the following steps:

1. ***Missing Relationships Detection***: To find all missing *inter-ontology relationships* we identify the resources that appear in multiple ontologies. If a resource (referred to as *"reused resource"*) is used in multiple ontologies (referred to as *"hosting ontologies"*) then there must be some *inter-ontology relationships*. If these rela-

tionships are not explicitly defined then, there are missing relationships among the ontologies.

2. ***Relationship Direction Identification***: Since *inter-ontology relationships* are directed links between ontologies, another challenge is to find the direction of the missing relationships. A part of the ontology corpus in Figure 6.2 (top right), contains a *reused resource* (i.e. node 'c') that appears in three different ontologies O', O'' and O'''. In the absence of explicit relationships, some implicit relationships exist and to create these relationships we need to identify the direction of the relationships, i.e. from O' to O'' and from O''' to O''. To identify the direction, the *namespace* of the *reused resource* are used. If the namespace of the *reused resource* matches to the *namespace* of a *hosting ontology* (e.g. O''), then the ontology is selected as the *"home ontology"* of the *reused resource* and the *inter-ontology relationships* are directed from the *hosting ontologies* (i.e. O', O''') to the *home ontology* i.e. O''.

3. ***Explicit relationships Creation***: Once the missing relationships and their directions are identified, we create explicit *inter-ontology relationships* using `owl:imports` properties.

The *inter-ontology relationship* extraction process is briefly described in Algorithm 1. Firstly the namespace of each ontology is identified (line 1-3). `TopNS(`o_i`)` returns the namespace that is the namespace of most of the resources in o_i. The SPARQL query to find the namespaces in an ontology and the count of resources defined with each namespace is shown in Listing 6.1. Secondly, all *reused resources* are identified and each resource and a corresponding list of *hosting ontologies* are recorded in M_{ro} as a key value pair (line 4-9). Finally, for each resource in the M_{ro} the *home ontology* is identified, and the resource URI is replaced with the ontology URI and all missing *inter-ontology relationships* for an ontology are recorded in M_{oo} (line 10-19). An important point to consider is that although an ontology *OntA* may reuse more than one resource from another ontology *OntB* there will only be one *inter-ontology relationship* from *OntA* to *OntB* according to the semantics of the *owl:imports* property. Therefore, independently of the number of resources that are reused in *OntA* from *OntB*, we create a single *inter-ontology relationship* from *OntA* to *OntB*.

Table 6.1 and Table 6.2 show the top five ontologies in the benchmark ontology collection [Butt et al., 2014a] and the corresponding number of *inter-ontology relationships* that are directed to these ontologies (i.e. *reuse count)* counted through *explicit* and *implicit* relationships, respectively. Top five reused ontologies based on implicit inter-ontology relationships are clearly the most popular ontologies in Semantic Web. A detailed analysis on the effectiveness of `FindRel` is presented in Section 6.5.2.4.

6.4 Online Query Processing

In this section, we first describe the concept retrieval task and then we outline the online query processing technique that finds the top-k ranked concepts for `Q` in `O`

Algorithm 1: FINDREL: Inter-Ontology Relationships Extraction

Input: A finite set $O = \{o_1, \ldots, o_n\}$ of Ontologies
Output: An Index M_{oo} that maps inLinks of all o_i

for $i \in [1, n]$ **do**
  $ns_{o_i} \leftarrow$ topNS(o_i);
  M_{ns}.put(o_i, ns_{o_i});
for $r \in o_{i\in[1,n]} \wedge M_{ro}.contains(r) = false$ **do**
  while $\exists\ o_{j\in[1,n]} : (r \in o_j) \wedge (o_i \neq o_j)$ **do**
    $oList_r$.add(o_j);
  if $oList_r.size() > 0$ **then**
    $oList_r$.add(o_i);
    M_{ro}.put(r, $oList_r$);
while $\exists\ r_{k\in[1,M_{ro}.size()]}$ **do**
  $ns_{r_k} \leftarrow$ getNS(r_k);
  for $s \in [\ 1, oList_{r_k}.size()]$ **do**
    $ns_{o_s} \leftarrow M_{ns}$.get($o_s$);
    if $ns_{o_s} = ns_{r_k}$ **then**
      $o_k \leftarrow o_s$
      break;
  if $M_{oo}.contains(o_k)$ **then**
    $oList_{r_k}$.addAllDistinct(M_{oo}.get(o_k))
  M_{oo}.put(o_k, $oList_{r_k}$)
return M_{oo}

with the highest semantic relevance.

6.4.1 Concept Retrieval Task

Given a query string $\mathcal{Q} = \{q_1, q_2, \ldots, q_k\}$, an Ontology corpus $\mathcal{O} = \{O_1, O_2, \ldots, O_n\}$ and a word sense similarity threshold θ, the `concept retrieval task` is to find the $C_Q = \{(v_1, O_1), \ldots, (v_i, O_j)\}$ from $\mathcal{O}$, such that there is a *surjective function* f_{sj} from $\mathcal{Q}$ to C_Q where (a) v has a partial or an exact matched word $\phi(q_v)$ for q $\in \mathcal{Q}$ (b) for a partially matched word, $SenSim(q\ ,\ \phi(q_v)) \geq \theta$. We refer to C_Q as a candidate set of Q introduced by the mapping f_{sj}.

$SenSim(q\ ,\ \phi(q_v))$ is a word similarity measure of a query keyword and a partially matched word in $L(v)$.

6.4.2 Query Evaluation

In the online query evaluation (cf. Figure 6.3), first a candidate set for a *top-k* concept is selected from the ontology data store i.e. *OntDataStore*, and then the relevance of each concept is calculated based on the formulae defined in Equation 6.6.

Listing 6.1: SPARQL query

```
SELECT
  ?namespace (count(?s) AS ?count)
FROM <aGraph>
WHERE {
  { {?s rdf:type owl:Class.}
    UNION
    {?s rdf:type rdfs:Class.}
  }
    Bind (REPLACE(str(?s),
    "[^/#]+$", "") AS ?namespace)
} Group By ?namespace
  ORDER BY DESC(?count)
```

Table 6.1: Top five reused ontologies based on explicit inter-ontology relationships

URI	Reuse Count
http://def.seegrid.csiro.au/isotc211/iso19150/-2/2012/basic	36
http://purl.org/dc/elements/1.1/	25
http://www.ifomis.org/bfo/1.1	16
http://www.w3.org/2006/time	16
http://www.ontologydesignpatterns.org/schemas/cpannotationschema.owl	15

6.4.2.1 Candidate Result Set Selection

A keyword query evaluation starts with the selection of a candidate set C_Q for Q. A candidate result set C_Q is characterised by two features:

1. To be part of the candidate set a candidate concept v must have at least one exact or partial match $\phi(q_v)$ for any query keyword $q \in Q$ as part of the value of (a) `rdfs:label` (b) `rdfs:comment` (c) `rdfs:description` property; or $\exists\, q \in Q \mid \phi(q_v)$ is part of *L(v)*.

2. The word sense similarity of q and $\phi(q_v)$ i.e. senSim(q,$\phi(q_v)$) should be greater than the sense similarity threshold θ.

In our current implementation, we check the word sense similarity using WordNet and set a word sense similarity threshold θ = 0.85. Each entry in a candidate list denotes a candidate concept *'v'* and is a pair *(v,O)* (shown in Figure 6.3) of v and O where $v \in V(O)$. Since for the *reused resources* there are multiple *hosting ontologies*, therefore *'v'* may have multiple entries in a candidate set if it is a *reused resource*.

Table 6.2: Top five reused ontologies based on implicit inter-ontology relationships

URI	Reuse Count
http://www.w3.org/2002/07/owl#	881
http://www.w3.org/2000/01/rdf-schema	361
http://www.w3.org/1999/02/22-rdf-syntax-ns	298
http://xmlns.com/foaf/0.1/	228
http://www.w3.org/2004/02/skos/core	140

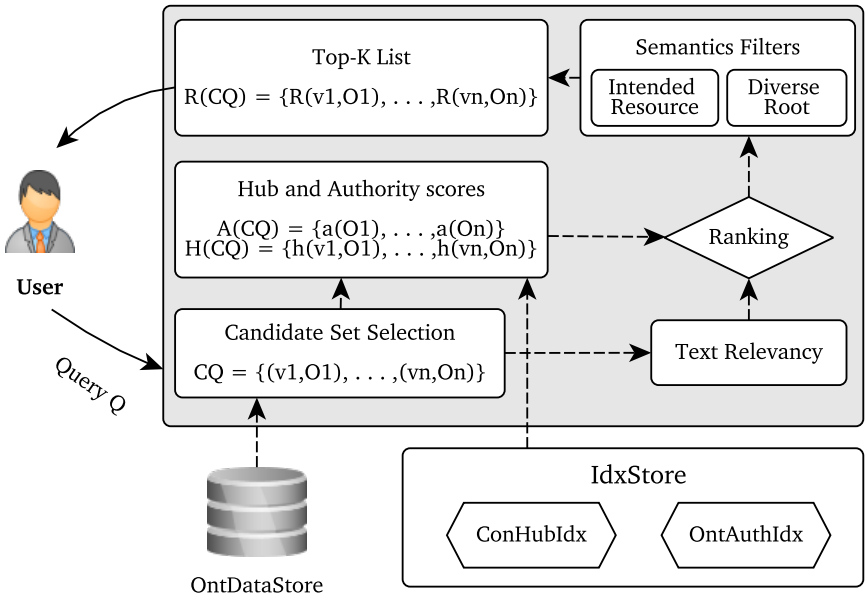


Figure 6.3: Online Query Processing

6.4.2.2 Concept Relevance

For each entry in the candidate list, two scores are retrieved from the stored indices built during the *offline ranking phase*. The entry *(v,O)* is used to retrieve the *hub score* of concept v in ontology O from the *ConHubIdx*, and the *authority score* of ontology O from the *OntAuthIdx*. The two scores are combined according to the formulae of Equation 6.6, that provides the final *concept relevance* of each v to the Query Q.

6.4.3 Filtering top-k results

In this section, we discuss the filtering strategies of our framework to enhance the semantic similarity of the results to the keyword query. We introduce two properties for the *top-k* results:

Diverse Results Semantic. Considering the semantics of a query allows us to re-

Algorithm 2: TOP-K FILTER

Input: Concept Relevance Map $R(C_Q)$ = {[(v_1, O_1),r_1], .. , [(v_n, O_n),r_n]}
Output: top-k results $L(C_Q)$ = {[(v_1, O_1),r_1], .. , [(v_k, O_k),r_k]}

1 $R_s(C_Q)$ /* A map to store intermediate results */
2 **for** $i \in [1, n]$ **do**
3 $e \leftarrow$ R(C_Q).get(i);
4 **if** *R(C_Q).contains(e')* $\bigcap$ *v(e)= v(e')* $\cap$ *O(e)* $\neq$ *O(e')* **then**
5 $R_s(C_Q)$.put($[(v, O_h), r_h]$);
6 **for** e'' ***where*** $v(e'') = v$ *and* $O(e'') \neq O_h$ **do**
7 $R_s(C_Q)$.put($[(v, O''), (r'' - r_h)]$);
8 R(C_Q).removeAll(e where concept is v);
9 **else**
10 $R_s(C_Q)$.put(e);
11 $R_s(C_Q) \leftarrow$ sortByValue($R_s(C_Q)$);
12 **while** *($L(C_Q)$.size() $\leq$ k)* $\bigcap$ *($i \in [1, n]$)* **do**
13 $e \leftarrow$ R(C_Q).get(i);
14 **if** *ϕ ($q_v(e)$) is a multi-keyword match* **then**
15 **if** $I_t(\phi\ (q_v(e))) = q$ **then**
16 $L(C_Q)$.put(e);
17 **else**
18 $L(C_Q)$.put(e);
19 **return** $L(C_Q)$

move repetitive results from the top-k results to increase the diversity in the result set. As mentioned earlier, if a candidate concept v is reused/extended in *'n' hosted ontologies* i.e. $\{O_1, O_2, ..., O_n\}$ then it may appear multiple times in a *candidate result set* (i.e. C_Q = {(v, O_1), (v, O_2),...,(v, O_n)}). In this case we remove the duplicates from the candidate result set.

Intended type Semantic. The semantic of a concept label differentiates the *intended type* from the *context resource* of a concept. The label of a concept v may have multiple keywords as a description of the concept, e.g., the label of a concept in the GND ontology has the keywords "Name of the Person"[3]. Here "Name" is the intended type, whereas "Person" is the context resource. According to the *intended type semantic property* a concept should appear in the *top-k* if and only if its *intended type* matches to at least one of the query keywords q $\in$ Q.

Algorithm 2 explains the *top-k results filtering* process. It takes as input a *Concept Relevance Map* $R(C_Q)$ and returns the *top-k* results. First, the *diverse results semantics* are preserved (line 2-10) for $R(C_Q)$, and then the check for *intended type semantics* is applied (line 11-18) until the *top-k* results are retrieved.

[3] http://d-nb.info/standards/elementset/gnd#NameOfThePerson

A map $R_s(C_Q)$ is initialised to store the intermediate results that preserve the *diverse results semantics*. All candidate concepts in $R(C_Q)$ that appear only once in $R(C_Q)$ preserve the *diverse results semantics*, therefore they become part of $R_s(C_Q)$ (line 10). For all *reused concepts*, first the *home ontology* $O_h(v)$ of the concept v is identified. The entry e=[(v, O),r] $\in R(C_Q)$ for which the ontology of the concept v is its home ontology (i.e. O=$O_h(v)$) becomes part of the $R_s(C_Q)$ (line 5). For all other entries e'' for v a new entry is created by subtracting the relevance score of e i.e. r_h from the r'' and add it to the $R_s(C_Q)$ (line 6-7). The process decreases the relevance score of duplicate entries by a factor of r_h. Then all such e'' from $R(C_Q)$ are removed since they have already been dealt with through candidate concepts of v.

The next step is to check the *intended type semantic*. For brevity, a detailed discussion of the intended type checking is exempted from Algorithm 2. The *ontology structure* and the *Information Retrieval methods* are used to identify the *intended type*. For a concept v, its sub-classes, super-classes and inter-ontology relationships are extracted as the context of v. The WS4J[4] API is used to calculate the similarity of different words in the concept v with its context. The word that has a higher similarity score in regards to the context is considered as the intended type of the concept. However, to reduce the cost of ensuring the *intended type semantic* for *top-k* results, the filter is only applied until we retrieved the *top-k* results in the final results L. For this, first the $R_s(C_Q)$ is sorted in a decreasing order based on its relevance score r, so the most relevant results for query Q are at the top of the $R_s(C_Q)$ (line 11). Then the intended type of the candidate concept is checked only until 'k' concepts are selected from $R_s(C_Q)$ or there are no more results in $R(C_Q)$ (line 12). If the concept v has a single exact or partially matched word ϕ $q_v(e)$ then by default it preserves the semantics and becomes part of $L(C_Q)$ (line 18); otherwise we check its *intended type*. If its intended type is equal to the query keyword $q \in Q$, the concept is included in $L(C_Q)$ otherwise, it is ignored.

6.5 Experimental Evaluation

In the following we present an experimental evaluation of our *relationship based top-k concept retrieval framework* on a benchmark suite CBRBench - Canberra Ontology Ranking Benchmark [Butt et al., 2014a]. We conducted three sets of experiments to evaluate: (1) the effectiveness of the *relationship based concept retrieval framework*, (2) the quality of *HubScore* presented in Section 6.3.1.1 and the (3) Effectiveness of *FindRel* algorithm.

6.5.1 Experimental Settings

To evaluate our approach we use a benchmark suite CBRBench [Butt et al., 2014a] developed above, that includes a collection of ontologies, a set of benchmark queries and a ground truth established by human experts. This collection is composed of

[4] https://code.google.com/p/ws4j/

1022 ontologies and ten keyword queries: `Person`, `Name`, `Event`, `Title`, `Location`, `Address`, `Music`, `Organization`, `Author` and `Time`. The benchmark evaluates eight state-of-the-art ranking algorithms on the task of ranking ontologies.

We use the performance of these ranking models as the baseline to evaluate our approach. For a fair analysis, we implemented two versions of our approach: (1) `DWRank`: the DWRank model with the *diverse root semantics* (2) `DWRank+Filter`: the DWRank model with both the *diverse root semantics* and the *intended type semantics*. The reasoning for having two different implementations of our top-k concept retrieval framework is, that we want to be able to compare the effectiveness of the DWRank model against the top-k results of the baseline ranking models of CBR-Bench - which means the *diverse root semantics* model of `DWRank` is considered as to be evaluated against the baseline ranking models. As the *intended type semantics* can be applied to any of the baseline ranking models to improve their performance, the `DWRank+Filter` can not be compared to the baseline ranking models and we only evaluate the effectiveness of the *intended type semantics filter* compared to `DWRank` without filters. The effectiveness of the framework is measured in terms of its *Precision* `(P)`, *Average Precision* `(AP)`, *Discounted Cumulative Gain* `(DCG)` and *Normalised Discounted Cumulative Gain* `(NDCG)`.

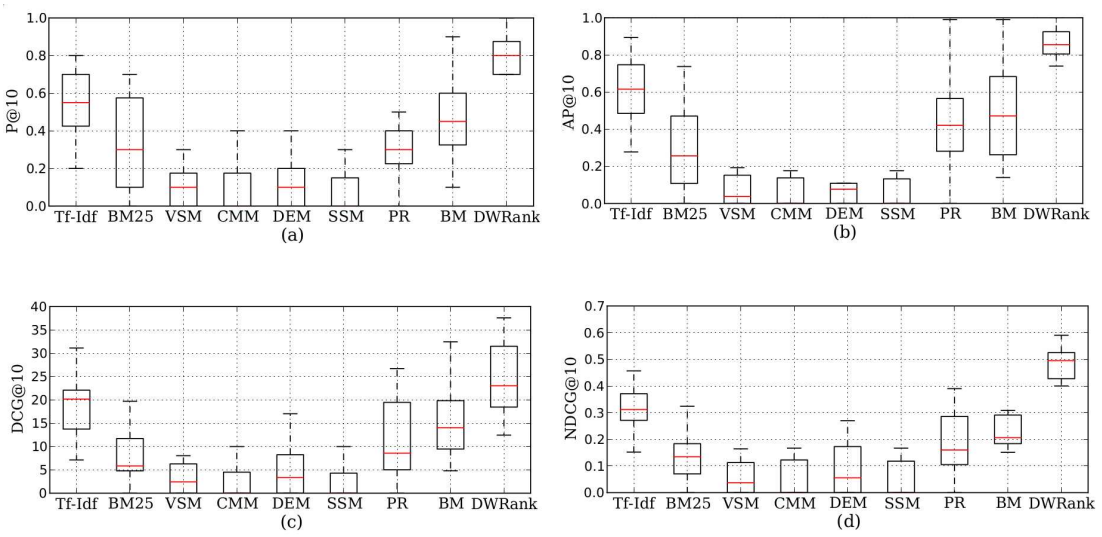


Figure 6.4: Effectiveness of Ranking Model (Tf-Idf = Term Frequency Inverse Document Frequency, BM25= BM25, VSM = Vector Space Model, CMM = Class Match Measure, DEM = Density Measure, SSM = Semantic Similarity Measure, PR = PageRank, BM = Betweenness Measure)

6.5.2 Experimental Results

We next present our findings.

6.5.2.1 Effectiveness of DWRank

In the first set of experiments, we evaluated the effectiveness of DWRank in comparison with the eight baseline ranking models. We ran the ten sample queries on the ontology collection and retrieved the *top-k* results according to the proposed ranking model. We recorded the P@10, the AP@10, the DCG@10 and the NDCG@10. The effectiveness measure results of the DWRank are shown in Table 6.4, where the column headers correspond to benchmark query terms and row headers correspond to the evaluation metrics.

Next, we compared our results with the baseline for the same dataset with the sample queries. The results are shown in Figure 6.4. Each graph here presents an effectiveness measure of a ranking model for all ten queries, where the x-axis is the *ranking model* and the y-axis is the *unit of measure*. Each box on a graph presents the range of effectiveness measure for 10 sample queries according to the gold standard. Figure 6.4 shows the maximum, minimum and average performance of DWRank in comparison to the performance of the baseline ranking models for each of the ten queries. The graph shows that DWRank performs better than the best performing ranking algorithm for most queries. For the *address* query, the P@10 and AP@10 for DWRank is lower than the other best-performing ranking model. However, the maximum average AP@10 for DWRank on ten queries is 0.84 that is greater than the average of Tf-Idf, the best baseline ranking models, (i.e., 0.55). The box plot also shows that P@10 and AP@10 of DWRank ranges from 0.7~1.0 that means the performance of DWRank is more stable on the ontology collection for the sample queries than the baseline ranking models.

Similarly, the DCG@10 values in Figure 6.4(c) and NDCG@10 values in Figure 6.4(d) for the ranking models show that DWRank is more effective than the baseline models. The maximum and minimum measures are closer to the *Betweenness Measure (BM)* and the *Tf-Idf* model, however, the average performance of DWRank is much higher than the average performance of the BM and Tf-Idf models.

Figure 6.5 compares the AP@10 (resp. NDCG@10) for DWRank on all ten queries with the maximum AP@10 (resp. NDCG@10) achieved with any of the baseline ranking model on the sample queries. The result shows that DWRank performs best for AP@10 (resp. NDCG@10) for all but one query. The experiment confirms our claim about the stable performance of the DWRank algorithm.

6.5.2.2 Effectiveness of DWRank+Filter.

For the evaluation of the filter performance, we ran the ten sample queries of the benchmark collection with the DWRank model extended with the filter proposed

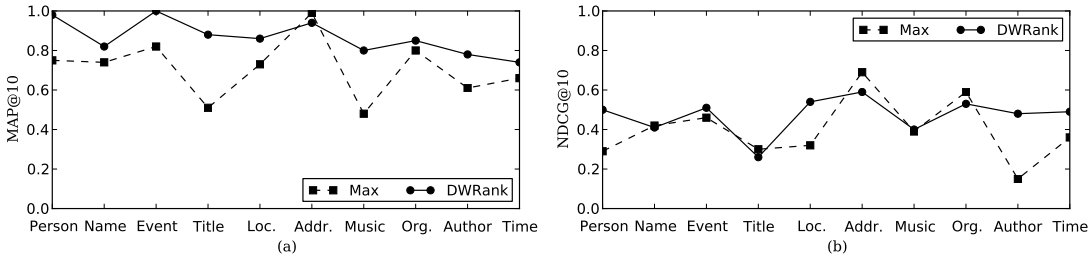


Figure 6.5: *AP@10* and *nDCG@10* for DWRank in comparison with the best value for any ranking model on sample queries

earlier, i.e. *intended type semantics*. Figure 6.6 shows the effectiveness of DWRank compared to DWRank+Filter. The average `P@10` increased from 0.8 to 0.9, i.e. a 12% increase in the effectiveness of the results.

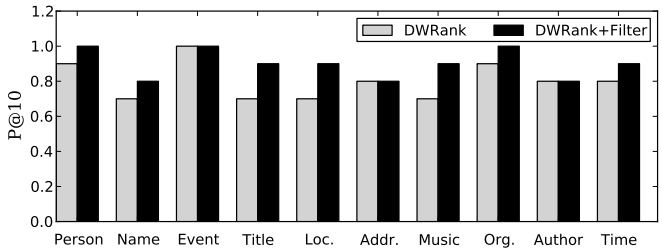


Figure 6.6: Filter Effectiveness

From the evaluation, it is obvious that the *filter* improves the overall performance of our framework. Some analysis on the precision and recall of the *filter* in terms of *True positive* (`TP`), *False positive*(`FP`), *True negative* (`TN`) and *False negative* (`FN`) examples regarding our current implementation of the intended type semantic filter are shown in Table 6.3. We analyse the top-10 results of DWRank without the *intended type semantic filter* and then with the filter. For each query if there are `TN`, `FN`, `FP` examples we selected them or otherwise a random `TP` example.

Table 6.3: Intended Type Semantic Filter Performance in Relationship-based top-k Concept Retrieval Framework

Query term	Label of concept	Human Judgement	Intended Type Filter Judgement
person	personal communication model	×	×
name	gene name	×	×
event	academic event	✓	✓
title	spectrum title	×	×
location	hematopoiesis location trait	×	×
address	E45_address	✓	×
music	sound and music computing	×	×
organization	3D structural organization datrum	×	×
author	author list	✓	×
time	time series observation	×	✓

Table 6.4: DWRank Effectiveness

	Person	Name	Event	Title	Location	Address	Music	Organization	Author	Time
P@10	0.9	0.7	1	0.7	0.7	0.8	0.7	0.9	0.8	0.8
AP@10	0.98	0.82	1	0.88	0.86	0.94	0.80	0.85	0.78	0.74
DCG@10	37.58	19.11	35.12	12.45	24.88	23.53	14.82	33.70	18.24	22.53
NDCG@10	0.51	0.41	0.51	0.26	0.60	0.59	0.4	0.53	0.48	0.49

Table 6.5: HubScore Quality: Centrality of Concepts

	Semantic Sensor Network Ontology (SSN)			ABS Ontology		
Rank	Reference Answers	HubScore	CARRank	Reference Answers	HubScore	CARRank
1	Sensor	**System**	**System**	Employer	**Employer**	**Person**
2	Observation	**Observation**	**Observation**	Person	**Period**	**EmployeeRole**
3	Property	**FeatureOfInterest**	**Deployement**	EmployeeRole	**Person**	**LegalEntity**
4	SensorOutput	**Deployement**	Platform	EmployerRole	**EmployerRole**	**EmployerRole**
5	SensorInput	Platform	SensingDevice	Period	**EmployeeRole**	Company
6	Stimulus	**SensorOutput**	**Sensor**	AusBusinessNo.	**LegalEntity**	**Employer**
7	FeatureOfInterest	**Sensing**	**SensorOutput**	Address	Company	PrivateCompany
8	Sensing	**Property**	Device	LegalEntity	**TypeOfActivityUnit**	PublicCompany
9	System	**Sensor**	**Sensing**	TypeOfActivityUnit	IncomePeriod	**TypeOfActivityUnit**
10	Deployement	Device	SurvivalRange	EnterpriseGroup	Occupation	AsicRegistration

Table 6.6: Representative Ontologies

Ontology	No. of Concepts
Australian Bureau of Statistics Ontology (abs.owl)	43
Semantic Sensor Network Ontology (ssn.owl)	51
Project Ontology (project.owl)	51
Science Ontology (science.owl)	83

6.5.2.3 Quality of HubScore - Measure of the centrality of a concept

To evaluate the quality of the *hub score* we consider CARRank [Wu et al., 2008a] as a baseline. The reason of comparing the *hub score* quality with the quality of CARRank is two-fold: (1) CARRank use a similar approach (i.e. ReversePage Rank), and (2) the performance results in [Wu et al., 2008a] prove it a better approach than other centrality measures e.g. Betweenness Measure [Alani et al., 2006] and Density Measure [Alani et al., 2006]. Since the CARRank algorithm and the gold standard are not available online, we implemented CARRank in Java and adopted a similar evaluation strategy as presented in [Wu et al., 2008a].

To evaluate the two approaches, we tried to collect ontologies and their top-10 concepts. Four representative ontologies where members of CSIRO were part of the ontology design team were selected as shown in Table 6.6. We asked the ontology creators of the four ontologies to list the top 10 central concepts of the ontology they designed. We then compare the reference ranking produced by the ontology creator with the top-10 ranked list generated by HubScore and CARRank. Table 6.5 presents the comparison on the concepts ranking for the SSN and the ABS ontology. Concepts listed in bold font are relevant ranking results.

In Table 6.5 `HubScore` ranks 8 (resp. 7) relevant answers in the top 10 ranking results for the SSN (resp. ABS) ontology in comparison to `CARRank` that ranks 7 (resp. 6) relevant answers for the SSN (resp. the ABS) ontology. Moreover, relatively more relevant results are ranked at the top of the list by HubScore. The quality of both these algorithms is measured in terms of P@10 for four representative ontologies and presented in Figure 6.7.

Though the precision of HubScore on the representative ontologies increases by 0%-20% compared to CARRank, the ranked list also seems more meaningful than CARRank. This can be seen in Table 6.7 that presents the top 5 concepts of the FOAF ontology ranked by HubScore and CARRank.

6.5.2.4 Effect of FindRel: Extraction of Implicit Inter-Ontology links

The Authority score calculation of DWRank in Section 6.3.1.2 is based on a link-based analysis (i.e. PageRank), that computes the popularity of an ontology in the ontology corpus. However, missing links among ontologies lead to wrong popularity

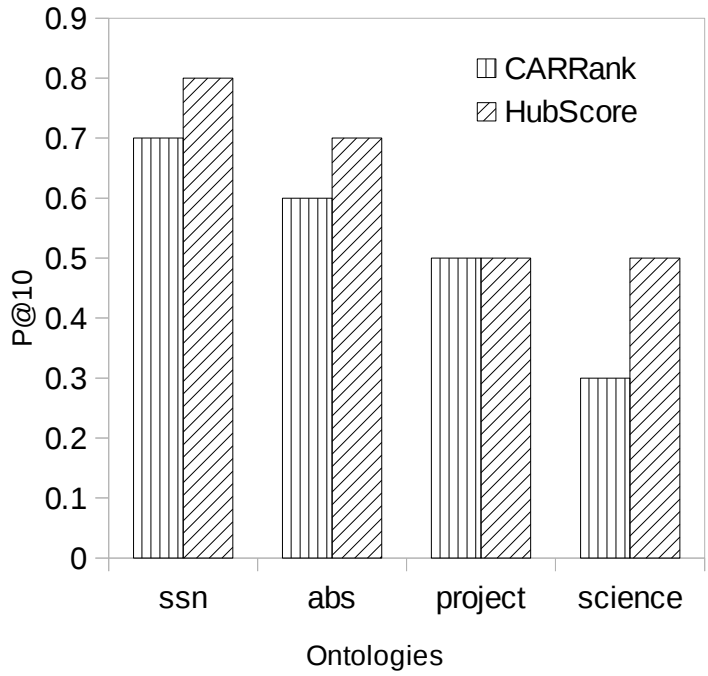


Figure 6.7: Comparison of Ranking Concepts

scores [Cheng and Qu, 2013]. We, therefore, find the missing *inter-ontology links* and present a graph-based analysis of the ontology corpus that shows the increased connectivity of the ontology corpus after extraction of the *implicit inter-ontology links* with `FindRel`.

Table 6.8 presents different statistical properties of the ontology corpus with and without considering explicit *inter-ontology relationships*. The `Node` notation represents the number of ontologies in the corpus. `Sink Node` is the number of ontologies that are imported (reused) by other ontologies and `Source Node` represents the number of ontologies that import at least one ontology. Whereas, `Isolated Node` counts the ontologies that neither import nor are imported by any other ontology in the corpus. Similarly, `Edge` is the count of links in the ontology corpus and `Average Degree` is the number of inlinks and outlinks for each node (ontology). `Highest Degree`, `Highest Indegree` and `Highest OutDegree` are the maximum number of in-links and outlinks, the maximum number of inlinks and the maximum number of outlinks for a node, respectively.

Three language-level vocabularies, namely RDF[5], RDFS[6] and OWL[7], and all the *inter-ontology links* involving them makes our statistics biased towards the improved results through the implicit link extraction. Therefore, they are excluded from the

[5] http://www.w3.org/1999/02/22-rdf-syntax-ns#
[6] http://www.w3.org/2000/01/rdf-schema#
[7] http://www.w3.org/2002/07/owl#

Table 6.7: HubScore Quality: Top-5 Concepts of Foaf Ontology

Rank	HubScore	CARRank
1	Person	Person
2	Agent	Agent
3	Group	OnlineGamingAccount
4	Organization	OnlineChatAccount
5	OnlineGamingAccount	OnlineEcom.Account

Table 6.8: Statistical Properties: Explicit vs. Implicit Inter-Ontology Links

	Explicit Link Graph	Implicit Link Graph
Node	1019	1019
Sink Node	177	348
Sink Node(%)	17.37	34.15
Source Node	204	815
Source Node(%)	20	79.98
Isolated Node	742	135
Isolated Node(%)	72.81	13.25
Edges	431	2311
Average Degree	0.85	4.54
Highest Degree	38	228
Highest Indegree	36	228
Highest Outdegree	26	29

following analysis.
`Nodes` remains the same in the explicit link ontology corpus and the implicit link ontology corpus, as only the missing links among ontologies are extracted in `FindRel`; however, they differ in the number of edges. As shown in Table 6.7, the `Implicit-Link Graph` contains more edges (links) than the `Explicit-Link Graph`. `Average Degree` and `Isolated Node` values for both the graphs represent that without *implicit inter-ontology links*, the ontology corpus is disconnected and most of the ontologies (i.e. isolated nodes) will end up with the same *authority score* that will minimise the effect of the authority score contribution towards the DWRank model. The `Implicit-Link Graph` statistics are more interesting because it shows that the meta-description of ontologies often fails to reflect the reuse of another ontology which exists in its description, indicating that meta-descriptions are not reliable in this respect. Therefore, it will be insufficient to only leverage meta-descriptions to perform the link analysis for carrying out tasks such as ranking.

6.6 Discussion

Our set of experiments showed that there are a number of factors that contributed towards the effectiveness of our approach. Firstly, we have evaluated the proposed algorithm, i.e, DWRank, against the state-of-the-art ranking models on a benchmark ontology collection. Our results showed that the characteristics of DWRank (centrality and authoritativeness) are effective enough to produce a good ranking. The results presented in Chapter 5 showed that popularity (i.e. reuse of ontology concepts) or graph based analysis (i.e. coverage or centrality) alone are not an effective model for ontology ranking. However, our approach leverages both, centrality (graph based approach) and reuse (popularity based approach) together to produce a ranking much closer to human expectations than the state-of-the-art ranking algorithms. Secondly, we evaluated the performance of the proposed filter, i.e. *intended type semantics*. The results showed that the proposed filters enhanced the effectiveness of DWRank, which implies that the filter, when used with any other ranking model, can improve the effectiveness of that model. Thirdly, we evaluated the effectiveness of the HubScore. Our experiments showed that although the precision of HubScore on the representative ontologies increases by 0%-20% compared to the baseline, the ranked list seemed more meaningful than the baseline. Finally, we evaluated the effect of finding missing inter-ontology relationships on the performance of the algorithm. The results were interesting because it showed that the meta-description of ontologies often fails to reflect the reuse of another ontology which exists in its description, indicating that meta-descriptions are not reliable in this respect. Therefore, it will be insufficient to only leverage meta-descriptions to perform the link analysis for carrying out tasks such as ranking.

Our proposed technique showed an improved performance when compared to the baseline in a number of ways. However, for some of the queries, the proposed algorithm failed to perform up to expectations. We believe that the underlying reason is the fixed weights used to combine the two proposed metrics. For optimal performance of an algorithm, the metrics' weights need to be reset for each user query [Alani et al., 2006]. However, setting the weights of metrics manually for each and every query is not a practical solution. One of the possible solutions is to solve the problem through the use of a machine learning approach. Therefore, rather than assigning fixed weights to each evaluation metric, in Chapter 7, we propose a model that uses a machine learning approach to learn the best weights to rank the search results for a query based on the experience.

6.7 Summary

In this chapter a relationship-based *top-k* concept retrieval and ranking framework is presented. The ranking model, proposed as part of his framework, is comprised of two phases, an offline ranking and index construction phase and an online query and evaluation phase. In the offline ranking phase, our DWRank algorithm computes a rank for a concept based on two features, the centrality of the concept in

the ontology, and the authority of the ontology that defines the concept. The online ranking phase filters the top-k ranked list of concepts. The evaluation shows that DWRank outperforms the best performing ranking algorithm for most queries while exhibiting a more robust performance. For most of the queries, Percision@10 is high when compared to baseline ranking models with an AP@10 of 0.84 that is higher than the AP@10 of the best performing ranking models of the benchmark i.e. 0.55. Although our algorithm shows significantly improved performance compared to the state-of-the-art in ontology ranking models, further improvements are possible. We next proposed an improvement to the DWRank by introducing learning to rank approach.

Chapter 7

Learning Concept Ranking for Ontology Search

In this chapter, we adapt a machine learning approach to rank concepts for ontology search. This contribution is an effort towards addressing the research question **RQ3** i.e., how to rank relevant resources and ontologies for keyword queries? First, we explain the need for a learning to rank approach for concept ranking in Section 7.1. Next, the learning concept ranking framework is presented in Section 7.2 and evaluation results are shown in Section 7.3.

7.1 Introduction

Most of the existing ontology retrieval approaches use either single evaluation metric or assign fixed weights to combine more than one evaluation metrics. [Butt et al., 2014a] shows that none of the commonly used evaluation metrics performs adequately. Moreover, for optimal performance of an algorithm, the metrics' weights need to be reset for each user query [Alani et al., 2006]. However, setting the weights of metrics manually for each and every query is not a practical solution. One of the possible solutions is to solve the problem through the use of machine learning approach. Therefore, rather than assigning fix weights to each evaluation metric, a machine learning approach is used to learn the best weights to rank the search results for a query based on the experience.

Machine Learning is a well-known technique used to perform predictions by utilising known information and knowledge. We have a model defined up to some parameters, and learning is the execution of a computer program to optimise the parameters of the model using the training data or experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data or both. Overall, machine learning uses existing data and knowledge to predict some currently unknown information. For supervised learning, in general, we have training data and a mathematical model that uses the training data to learn the parameters of the model. For a given machine learning problem, we also need to define a set of features that are specifically designed for a given problem. From the training data, values of the defined features are extracted and are utilised by the mathemati-

cal model to learn the values of the parameters. To evaluate machine learning algorithms, we also need test data where we extract values for the same set of features and let the learned mathematical model make the prediction by utilising the learned parameter values. *Learning to rank* is a machine learning technique for training the model in a ranking task. Learning to rank has been successfully employed in a wide variety of applications in Information Retrieval (IR), Natural Language Processing (NLP), and Data Mining (DM). Typical applications include document retrieval, expert search, definition search, collaborative filtering, question answering, keyphrase extraction, document summarization, and machine translation [Li, 2014].

In this chapter, we use learning to rank approach to enhance the effectiveness of concept ranking models. The ranking metrics, i.e. *Text relevancy*, *Hub Score* and *Auth Score*, defined in Chapter 6 are extended and used to learn a ranking model that combines these measures in a more effective way. The effectiveness of the proposed approach is measured by comparing the ranking produced by the proposed approach with DWRank presented in Chapter 6. Moreover, a comprehensive comparison of the proposed approach with state-of-the-art ranking models is also presented. The evaluation results show that the ranking produced by the proposed approach is more effective as compared to the baseline ranking models on CBRBench ontology collection and benchmark queries.

7.2 The Ranking Framework

This section describes how Learning to Rank (LTR) has been adapted to the concept ranking, followed by the overview of the learning concept ranking framework.

7.2.1 Learning to Rank

Learning to Rank is a machine learning technique used to induce a ranking model from the training dataset.

For a given ontology collection O, let $Q = \{Q_1, Q_2, \ldots, Q_n\}$ be the set of queries. For each query Q_i, let $C(Q_i) = \{{c_1}^i, {c_2}^i, \ldots, {c_n}^i\}$ be the set of relevant concepts to Q_i. We define a feature set as F = $(f_1, f_2, \ldots, f_m)$ where f_j is the function $f_j : C(Q_i) \rightarrow \mathbb{R}$, which assigns a real value as a relevance of each answer ${c_k}^i$ to a query Q_i. $f_j(c)$ is referred as feature of the concept c. A target feature f_t is a special feature, which determines the correct ordering of the concept. It is the ordering of the concepts based on the target feature f_t, which is obtained using the LTR approach.

Our purpose is to construct a ranking model 'M' to provide the most relevant results by automatically determining the weights of features in F. To learn the ranking model from different features in a quantitative metric, we need a set of training instances. Training instances can be regarded as past query experiences, which can teach the system how to rank the results when new queries arrive. Each training instance is composed of a query, one of its relevant/irrelevant answer and a list of features. Intuitively we want to identify the model 'M' from features that can rank relevant answers as high as possible for a given query in training set 'T'. We want a

model such that the rank score of relevant matches is maximised. The design choices we made to learn the ranking model are as follows:

7.2.1.1 Feature Set

Most of the existing approaches have used `frequency-based features` obtained by counting different patterns in RDF graphs or counting the number of occurrences in web search results, and `centrality-based features` obtained by applying graph theoretic algorithms like PageRank or HITS on the RDF graph [Dali et al., 2012]. However, as shown in Chapter 5, none of the evaluation metric alone performs the best for all type of queries, and *HubScore* and *AuthorityScore* explained in Section 6.3.1.1 and 6.3.1.2 respectively showed an improved performance as compared to other state-of-the-art features or matrices discussed in Section 5.2. Therefore, as a feature set we extend the core features of DWRank, i.e. *text relevancy*, *hub score* and *authority score*. A detailed description of the feature set is given below:

- **`Hub score`** $h(v, O)$**:** For a concept v, the `hub score` for v in O is the measure of centrality for the concept v in O and $h(v, O)$ is computed according to the method discussed in Section 6.3.1.1.
- **`Max_hub score`** $h_{max}(v, O)$**:** For a concept v, the `max_hub score` for v is the maximum hub score of any concept v' in the ontology where $v \in V(O)$ and $v' \in V(O)$.
- **`Min_hub score`** $h_{min}(v, O)$**:** For a concept v, the `min_hub score` for v is the minimum hub score of any concept v' in the ontology where $v \in V(O)$ and $v' \in V(O)$.
- **`Normalized Auth score`** $a(O)$**:** For a concept v, the `normalized auth score` for v is the measure of authoritativeness of concept v where $v \in V(O)$ and $a(O)$ is computed according to the method discussed in Section 6.3.1.2.
- **`Text relevancy`** $F_V(v, Q)$**:** For a concept v, the `text relevancy` of v to Q is the measure of matched query terms with v. $F_V(v, Q)$ is computed according to the method discussed in Section 6.3.1.3.

7.2.1.2 Training Data

To the best of our knowledge no training data set (gold standard data) is available for learning to rank ontology concept. As an alternative, training dataset can be generated from the ground truth published as part of CBRBench [Butt et al., 2014a]. The process of generating training dataset from CBRBench is presented in Figure 7.1. The benchmark[1] provides manually created relevance judgements for sample queries on the benchmark ontology collection. In this gold standard, each query has

[1] https://zenodo.org/record/11121#.VDcYdK3I9yA

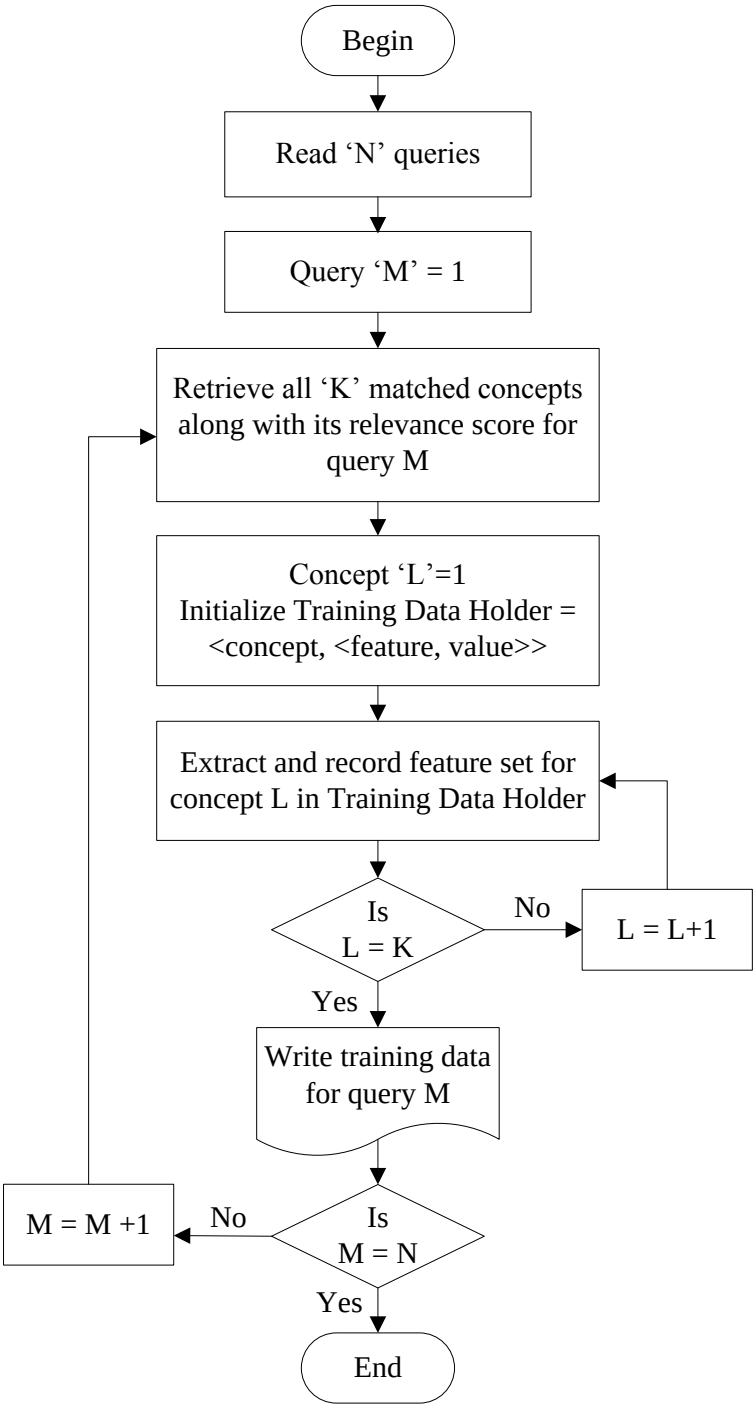


Figure 7.1: Generating training dataset for the proposed Learning Concept Ranking Framework

a list of relevant concepts and the relevance scores (0-4) of the concept to the query. For each query, we selected all relevant concepts along with their average relevance score as provided in CBRBench gold standard. We considered the average relevance score of a concept as the target feature. Moreover, for each concept a feature set is extracted using the *ConHubIdx* and the *OntAuthIdx* that are created in Section 6.3.1, and by computing the text relevancy score of that concept for the very query. A training dataset containing concepts and corresponding feature set is generated for each query. The file format for the training data, as well as testing and validation data, is the same as for SVM-Rank [Joachims, 2002]. The following lines represents one training example and is of the following format:

<target> qid:<qid> <$feature_1$>:<value> <$feature_2$>:<value> ... <$feature_n$>:<value> # <info>

Here `<target>` represents the value of target feature and it is a positive integer. In our case our feature target value is the rounded value of average relevance score of a concept to the query. The target score ranges from 0-4 for $'Irrelevant'$, $'SlightlyRelevant'$, $'Relevant'$, $'Useful'$, and $'ExtremelyUseful'$ respectively. `<qid>` is a positive integer that represents the query id of the query. <$feature_i$> is the feature number and `<value>` is a float that shows the feature value of the concept. `<info>` is any relevant information (a String) that is beneficial for general understanding but everything after # is ignored by the ranking algorithm. The training files general format in our case is as follow:

<relevance score> qid:[0-9] 1:<value> 2:<value> 3:<value> 4:<value> 5:<value> # conceptURIs

Query	Query Id
Person	0
Name	1
Event	2
Title	3
Location	4
Address	5
Music	6
Organization	7
Author	8
Time	9

Table 7.1: CBRBench Queries and their corresponding query IDs in training dataset

Feature	Feature Id
Text Relevancy	1
Hub Score	2
Max Hub Score	3
Min Hub Score	4
AuthScore	5

Table 7.2: Learning concept ranking framework features and their corresponding feature IDs in training dataset

For this purpose, we assign a query id (resp. feature id) to the benchmark queries (resp. features). The detail on the query id and feature id is shown in Table 7.1 and 7.2.

The dataset generated in this section is further divided into three partitions.

- *Training set:* a set of previous experiences used for learning: to fit the parameters of the ranking model, we would use the training set to find the 'optimal' weights.
- *Validation set:* a set of previous experiences used to tune the parameters of the ranking model.
- *Test set:* a set of examples used only to assess the performance of a fully-trained ranking model.

7.2.1.3 Metrics:

The ranking model is optimized for `Normalized Discounted Cumulative Gain (NDCG)` while training the model. The model is then tested against `Precision`, `Mean Average Precision (MAP)` and `Discounted Cumulative Gain (DCG)` metrics.

7.2.1.4 LambdaMART

A supervised machine learning approach is used to find the most appropriate concepts for user defined keyword queries. Learning to rank has been extensively studied in the machine learning community. We use `LambdaMART` [Wu et al., 2008b], a listwise learning to rank algorithm, which means that a list of training example of resources is provided where it is known which of the resource should be ranked higher in the result set. To learn the ranking model by the `LambdaMART` algorithm we use the RankLib[2] Library that provides an implementation of several learning to rank algorithms including `LambdaMART`.

7.2.2 Overview of the framework

In this section, we describe the learning concept ranking model to find the most relevant concepts for user queries. An overview of the framework is shown in Figure 7.2.

7.2.2.1 Index Construction and Learning Phase

During the index construction and learning phase, the proposed approach constructs the *ConHubIdx* and the *OntAuthIdx* on ontology corpus `O` as discussed in Section 6.3. The *ConHubIdx* maps each concept of an ontology to its corresponding *hub score*. Similarly, the *OntAuthIdx* maps each ontology to its precomputed *authority score*. The *hub score* and *authority score* are defined in Section 6.3.1. In next step, the training dataset is generated using the gold standard as described in Section 7.2.1.2. Since our approach uses CBRBench ontology collection as *Ontologydatastore*, we obtained feature sets (other than 'text relevancy') from the *ConHubIdx* and the *OntAuthIdx*

[2] http://sourceforge.net/p/lemur/wiki/RankLib/

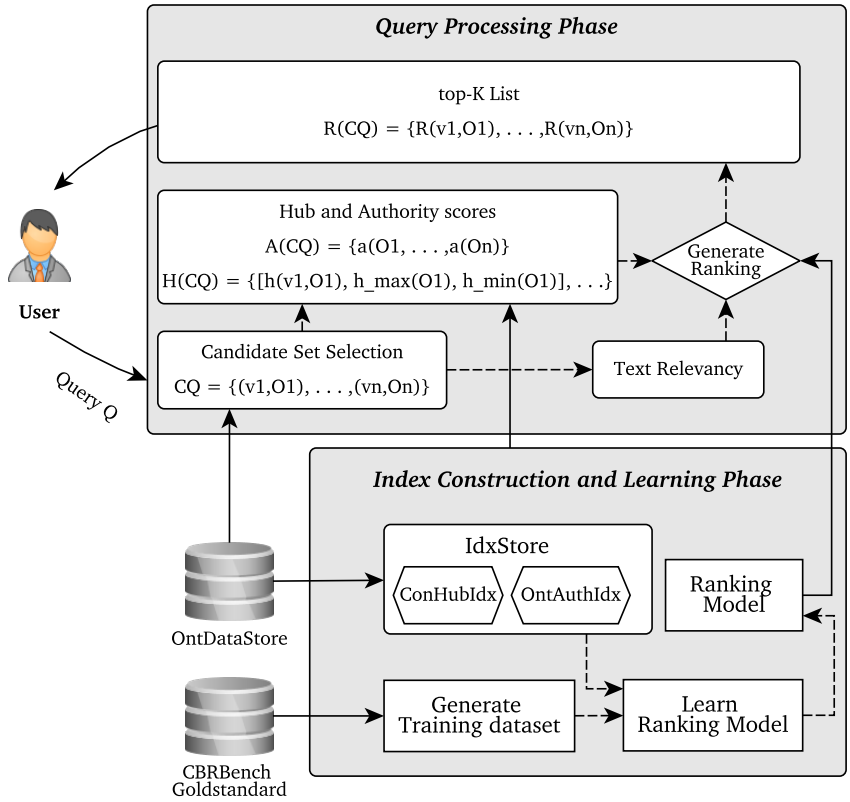


Figure 7.2: Online Query Processing

indexes. Next, a ranking model is learnt using a training dataset and LambdaMART. The model is then saved to produce a ranking for future queries.

7.2.2.2 Query Processing Phase

Upon receiving a query Q, the algorithm first extracts the *candidate result set* $C_Q = \{(v_1, O_1), ..., (v_i, O_j)\}$ including all matches that are semantically similar to Q by querying the ontology repository. The feature's values for each candidate result $(v, O) \in C_Q$ are extracted including *text relevancy*, *hub score*, *max hub score*, *min hub score* and *authority score* from the corresponding indices as $H(C_Q)$ and $A(C_Q)$ lists. The relevance scores for all candidate results are produced using the ranking model that is learned in the Index construction and learning phase. A ranked list $R(C_Q)$ of a candidate result set is computed by ordering the concepts in an descending order of their relevance score. $R(C_Q)$ is returned to the user as the most relevant results for a user query.

7.3 Evaluation Framework

In the following, we present an experimental evaluation of our *concept retrieval framework* on a benchmark suite, i.e. the CBRBench - Canberra Ontology Ranking Benchmark [Butt et al., 2014a]. We conducted a set of experiments to evaluate the performance of proposed approach.

7.3.1 Experimental Settings

For our experiments, we use our previously established CBRBench [Butt et al., 2014a]. It contains an ontology collection, benchmark queries, and a gold standard for the benchmark queries as shown in Chapter 5.

The effectiveness of the framework is measured in terms of its *Precision* (P), *Average Precision* (AP), *Discounted Cumulative Gain* (DCG) and *Normalised Discounted Cumulative Gain* (NDCG).

7.3.2 Experimental Results

7.3.2.1 Experiment-1: Effectiveness of Learning to Rank Approach

In this experiment, we study the impact of the learning to rank approach on the quality of the ranking. For the evaluation we implement two versions of DWRank:

1. DWRank Fixed Weight Linear Model: where hub score, authority score and text relevancy are combined in a linear model (i.e. Equation 6.6) and the values of weights α , β and γ are set to 0.5, 0.5 and 1 respectively.

2. DWRank with Learning to Rank Approach: By using LambdaMART, a LTR algorithm, a ranking model is learnt from the hub score, the authority score and the text relevancy along with two deduced features i.e. the max_hub score and the min_hub score.

For DWRank fixed weight linear model we run the ten sample queries on the ontology collection and retrieve the *top-k* results according to the proposed linear ranking model in Equation 6.6. We record the P@10, the AP@10, the DCG@10 and the NDCG@10. The effectiveness measure results of this implementation of DWRank are shown in Table 7.3. To evaluate DWRank with learning to rank approach, the Leave-one-out Cross Validation (LOOCV) approach is adopted as follows: for n number of queries we remove the relevance judgement for the training examples of one query and train the ranking model on the training examples of the remaining $n-1$ queries and then we evaluate the performance of the trained model on the n_{th} query. Once the process is repeated for n queries, the mean performance is computed. We apply LOOCV on the queries and the gold standard; and record the P@10, the AP@10, the DCG@10 and the NDCG@10. The results are presented in Table 7.3.

Table 7.3: Learning to Rank Concept Framework Effectiveness

	DWRank Fixed Weight Model				DWRank with LTR			
Query Terms	P@10	AP@10	DCG@10	NDCG@10	P@10	AP@10	DCG@10	NDCG@10
Person	0.9	0.98	37.58	0.51	1.0	0.8762	55.2391	0.8105
Name	0.7	0.72	19.11	0.41	0.6	0.6761	17.3283	0.4243
Event	1.0	1.0	35.12	0.51	1.0	0.8049	33.5500	0.6868
Title	0.7	0.78	12.45	0.26	1.0	0.9299	17.1655	0.4874
Location	0.7	0.86	24.88	0.60	1.0	0.8831	26.5708	0.5419
Address	0.8	0.89	23.53	0.59	0.9	0.8687	19.4308	0.6109
Music	0.7	0.80	14.82	0.40	0.7	0.7823	14.8165	0.6135
Organization	0.9	0.85	33.70	0.53	1.0	0.8902	56.7080	0.8321
Author	0.8	0.78	18.24	0.48	0.6	0.6725	16.7875	0.8129
Time	0.8	0.74	22.53	0.49	0.5	0.6601	14.5987	0.3907
Average	0.8	0.84	24.196	0.49	0.82	0.8044	27.2195	0.6211

The experimental results show that DWRank with the hub score, the authority score and the text relevancy combined with a model learnt through LTR performs better than the DWRank fixed weight linear model.

7.3.2.2 Experiment-2: Effectiveness of Top-k Search

Next, we compared our results with the available baseline for the sample queries. We compare the performance of the `DWRank fixed weight linear model` (so onward referred as DWRank) with the baseline algorithms. The results are shown in Figure 7.3. Each graph here presents an effectiveness measure of a ranking model for all ten queries, where the x-axis is the *ranking model* and the y-axis is the *unit of measure*. Each box on a graph presents the range of effectiveness measure for 10 sample queries according to the gold standard.

Figure 7.3 shows the maximum, minimum and average performance of DWRank in comparison to the performance of the baseline ranking models for each of the ten queries. The graph shows that DWRank performs better than the best performing ranking algorithm for most queries. For some of the queries, the `P@10` and `AP@10` for DWRank is lower than the other best performing ranking models. However, the maximum average `AP@10` for DWRank on ten queries is 0.80 which is greater than the average of Tf-Idf, the best baseline ranking model, (i.e., 0.55). The box plot also shows that `AP@10` of DWRank ranges from 0.65 ~1.0 that means the performance of DWRank is more stable on the ontology collection for the sample queries than the baseline ranking models.

Similarly, the `DCG@10` values in Figure 7.3(c) and `NDCG@10` values in Figure 7.3(d) for the ranking models show that DWRank is more effective than the baseline models.

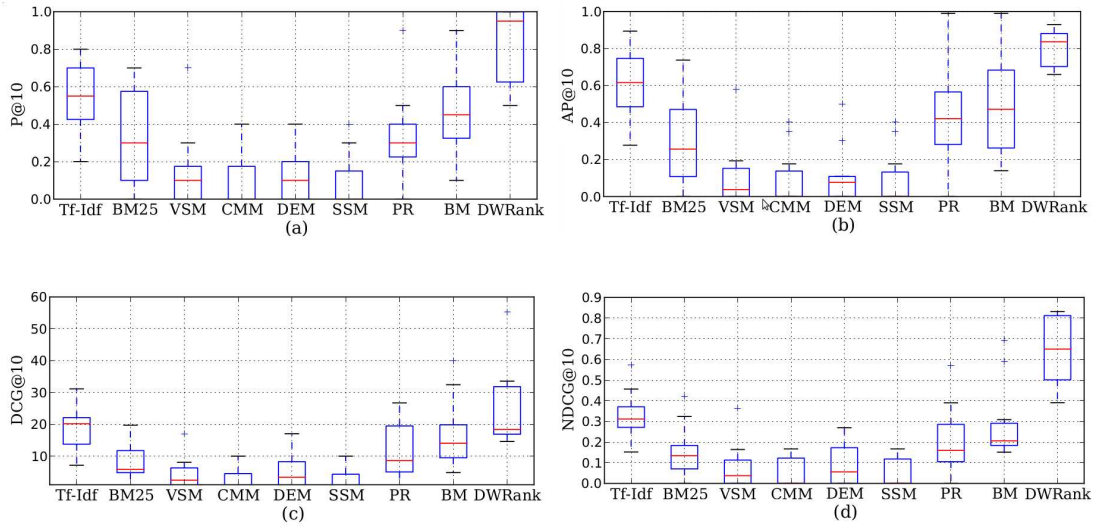


Figure 7.3: Effectiveness of Ranking Model(Tf-Idf = Term Frequency Inverse Document Frequency, BM25= BM25, VSM = Vector Space Model, CMM = Class Match Measure, DEM = Density Measure, SSM = Semantic Similarity Measure, PR = PageRank, BM = Betweenness Measure)

The maximum and minimum measures are closer to the *Betweenness Measure (BM)* or the *Tf-Idf* model, however, the average performance of DWRank is much higher than the average performance of the BM and Tf-Idf models.

7.4 Discussion

The results show that `DWRank with learning to rank approach` outperforms `DWRank fixed weight linear model` as well as the baseline ranking models. Moreover, the results presented in Chapter 6 show that `DWRank fixed weight linear model` is more effective as compared to the baseline ranking models. This implies that although the learning to rank approach increases the effectiveness of the base model, i.e. DWRank, the characteristics of DWRank (centrality and authoritativeness) are effective enough to produce a good ranking. The results presented in Chapter 5 show that popularity (i.e. reuse of ontology concepts) or graph based analysis (i.e. coverage or centrality) alone are not an effective model for ontology ranking. However, our approach leverages both, centrality (graph based approach) and reuse (popularity based approach) together to produce a ranking much closer to human expectations than the state-of-the-art ranking algorithms.

However, the best practice for learning a ranking model is to employ a query log generated by real users. Since we did not have a real-world query log at hand, the ranking model is learnt using the gold standard available as a part of CBRBench. As future work, the performance of the proposed ranking model can be improved by learning the ranking model using a real query log in the form of a larger training

dataset. The techniques can be investigated to improve learning the ranking model in the absence of a real query log. One of the potential solutions is incremental learning, where the initial model is learnt using a human created gold standard and then incrementally improving the performance of the ranking model with the future real-world query log as the users start querying through the proposed search engine.

7.5 Summary

In this Chapter, we proposed a concept ranking framework that adapts a machine learning approach for the automatic selection of metrics (i.e. hub score, auto score and text relevancy) weights to achieve the better ranking quality of retrieved results. We defined the feature set, generated the training data set, and used LambdaMART to learn a ranking model. The learnt model is then used to find the ranking of user queries. We evaluated the proposed framework in comparison to the DWRank presented in Chapter 6 and state-of-the-art ranking models discussed in Chapter 5. Chapter 6 and 7 addressed the challenge of ranking concepts for keyword queries. In the next chapter, we propose an ontology recommendation framework that helps users in finding the best matching ontologies to a multi-keyword query.

Chapter 8

Ontology Recommendation for Structureless Queries

In this chapter, we address the research question **RQ4** i.e., "How to find the most relevant resources and ontologies that cover one or more resources users are interested in?" In this regard, we introduce *RecOn*, a framework that helps users in finding the best matching ontologies to a multi-keyword query. We first present preliminaries and outline the *RecOn* workflow in Section 8.2. Next, we present the ontology recommendation approach in Section 8.3. Section 8.4 reports on the evaluation results and Section 8.5 concludes the chapter.

8.1 Introduction

Ontologies are a shared conceptualization of knowledge in a specific domain of discourse. However, only if an ontology is reused and thus its conceptualization validated by others it becomes truly a shared conceptualization. The process of reusing existing ontologies is also cost-effective and produces high-quality conceptualization because referring to an established ontological term in another domain of discourse builds an interlinked model of conceptualizations with strong formal semantics. It also facilitates data interoperability on both the syntactic and the semantic level. The growth of available ontologies in vertical domains such as bioinformatics, e-commerce and the internet-of-things highlights an increasing need for ontology search, which is the process of finding ontologies for users' defined queries from an ontology collection. To find the relevant ontologies various terms within that ontology, such as classes and properties, are searched and matched to the queries. However, it is often difficult to find the right ontology for a given use case. A user may not know the exact classes or properties and their positions in an ontology (ontological structure) she wants, but requires that the ontology contains a set of resources as its constituents. To mitigate the problem, a schema-less and structure-less keyword-based query is commonly used for ontology search. The problem here is that it is still hard to choose between ontologies that match to such a keyword query. Further, if there is no exact match for a given query string, a subset of the query may be used to find ontologies of interest. However, considering ontology matches for all

subsets of the query terms results in a significant number of matches. Consequently, it is often too time-consuming for a user to explore the matched ontologies to find the most suitable one.

Some previous work [Alani et al., 2006; Noy et al., 2009; Noy and d'Aquin, 2012] has tackled the problem of finding and recommending ontologies. More recently, a dedicated ontology search engine has emerged [Vandenbussche and Vatant, 2014]. Some of the search engines (e.g. [Ding et al., 2004]) adopt document ranking algorithms to introduce ranking to their search results; most consider the popularity of terms in the ontology corpus. For this they often use the PageRank algorithm as the ranking factor, which although effective in some cases, as [Butt et al., 2014a] showed, hinders the visibility of newly emerging, but well-defined ontologies. Moreover, most of the ontology search systems retrieve ontological terms (concepts and relations) and only a few provide ontology search based on a keyword query. Only a few ontology libraries and search engines facilitate the task of ontology retrieval for a user who is looking for an ontology that models all or some of the concepts she is looking for. The National Center for Biomedical Ontology (NCBO) proposed a biomedical ontology recommender web service [Jonquet et al., 2010] that is one of the most prominent approaches to find an ontology based on the text description. It is also a domain dependent ontology library and does not deal with all types of ontologies. A general solution is required for ontology search based on text descriptions or at least a multi-term query string.

`RecOn`[1] , an Ontology Recommendation approach, is an effort towards a dedicated ontology search engine that recommends relevant ontologies in response to a multi-term query string. Given a keyword query Q and a partial match approach, one might find many matches of Q in an ontology corpus. Thus, a user-friendly ontology search engine must address the following two questions: (1) how to determine which match is better, and (2) how to identify the top k matches? We propose an ontology recommendation approach that first finds the matched (relevant) ontology set to a query string; and then identifies the up to k most relevant ones. To identify the k most relevant ontologies for a query string, three measures are computed for each ontology: *matching cost* - the syntax and structural difference of the ontology from the query, *informativeness* - the information an ontology contains about the concepts that match the query string and *popularity* - the popularity of the ontology in the ontology corpus. We then find the relevance of an ontology to the query by formulating and solving ontology recommendation as a linear model, referred to as $RecOn_{ln}$, and as an optimisation problem mentioned as $RecOn_{opt}$. The aim is to find the ontologies that are as informative and popular as possible while incurring the least matching costs. The approach is evaluated on the CBRBench dataset [Butt et al., 2014a] against *AKTiveRank* by conducting a user study. The results of our user study show that $RecOn_{opt}$ and $RecOn_{ln}$ outperforms the state-of-the-art baseline algorithm *AKTiveRank*; and $RecOn_{opt}$ is efficient as well as effective as compared to $RecOn_{ln}$ on CBRBench ontology collection and sample queries designed in this work.

[1] A demo based on CBRBench ontology collection is available at www.activeraul.org/RecOn/, and the code is available at https://github.com/anilabutt/RecOn

8.2 RecOn: Ontology Recommendation

In the following, we first define the terms used throughout the chapter and then give a brief overview of the ontology recommendation workflow.

8.2.1 Preliminaries

Figure 8.1 and Figure 8.2 introduce a motivating example ontology and a sample query that are used throughout this chapter.

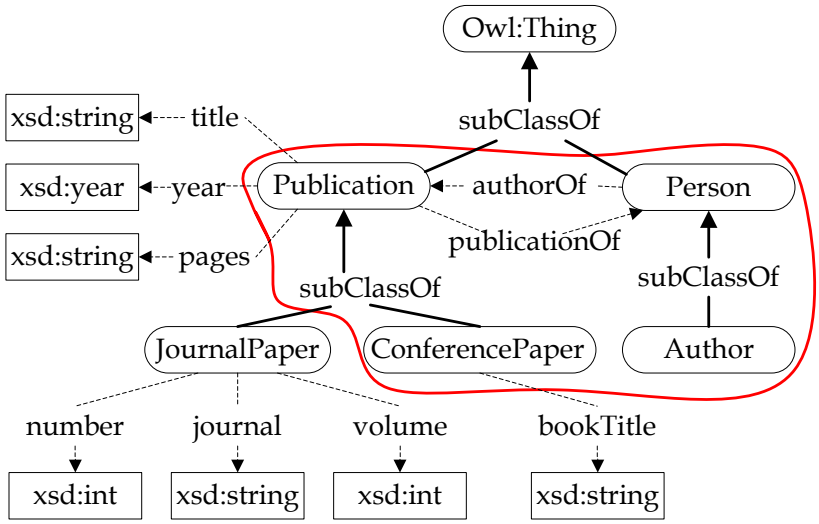


Figure 8.1: An Example Ontology

An ontology here refers to a labelled directed graph based formalisation $o = (C, R, L)$ of a domain knowledge. $C(o)$ is a finite set of nodes where $c \in C(o)$ denotes a domain concept in 'o' e.g., 'Publication' or 'Conference'. $R(o)$ is the set of edges where $r(c_i, c_j) \in R(o)$ denotes a relationship between c_i and c_j e.g., author(Publication,Person). L is a labelling function which assigns a label $L(c)$ (resp. $L(r)$ or $L(o)$) to node c (resp. an edge $r \in R(o)$, or the ontology 'o'). In practice, the labelling function L may specify (1) the node labels to relate the node to the referred concept, e.g. `'Person'`, `'Publication'` and `'Author'`; and (2) the edge labels as explicit relationships between/of concepts e.g., `'publicationOf'`, and `'title'` or implicit relationships e.g., `'subClassOf'` and `'superClassOf'`, and (3) the ontology label to relate the ontology to the domain or some identity.

Based on the description above we define the following functions:

- $l_c : C \rightarrow L(C)$ returns the label of concept 'c'
- $l_r : R \rightarrow L(R)$ returns the label of relation 'r'
- $domain(r) : R \rightarrow C$ returns the source concept/s of relation 'r'

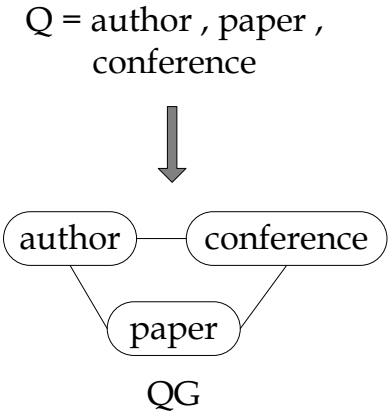


Figure 8.2: Query 'Q' and Query Graph 'QG'

- $range(r)$: $R \rightarrow C$ returns the target concept/s of relation 'r'
- $super(c)$: $C \rightarrow C$ returns the immediate super concept/s of concept 'c'
- $sub(c)$: $C \rightarrow C$ returns the immediate sub concept/s of concept 'c'

Table 8.1: Functions with outputs

Function	**Output**
l_{Person}	Person@en
$l_{authorOf}$	authorOf@en
$domain(authorOf)$	Person
$range(authorOf)$	Publication
$super(Author)$	Person
$sub(Publication)$	ConferencePaper, JournalPaper

Table 8.1 presents the results of the execution of these functions on the example ontology shown in Figure 8.1. Moreover, we define some terms on graph 'g', applicable to all graph based formalisations (e.g., ontology and query graph) used throughout this chapter, while Table 8.2 summarises the notations used for these terms.

Definition 1 : Concept Subsumption ($\subset_C$) . A concept set C_1(g) is subset of another set C_2(g) if every node (i.e., concept) in C_1(g) is in C_2(g). C_1(g) may have exactly the same nodes (i.e., concepts) as C_1(g).

$$C_1(g) \ \subset_C \ C_2(g) \ \text{iff} \ \forall \, c, \ c \in C_1(g) \ \rightarrow \ c \in C_2(g).$$

Table 8.2: Notations used throughout the chapter

Variable	Description
o	An ontology in Ontology Collection 'O'
op_i	Ontology pattern in $o_i : C(op_i) \subset_C C(o_i)$ and $R(op_i) \subset_R R(o_i)$
Q	Query String
q_i	Single query term in Q (i.e. $q_i \in$ Q)
QG	Query Graph w.r.t. Q
c_q	A query concept in Query Graph: $c_q \in C(QG)$
Q_{mat}	Match of QG in o
$\phi(c_q)$	$c \in C(Q_{mat}) : c \approx c_q$
$C_{Q_{mat}}$	Concept Match Set: $C_{Q_{mat}} = \{\phi_i(c_q) : \phi_i(c_q) \in C(Q_{mat})$ and $c_q \in C(QG)\}$

In Figure 8.1, if $C_1(o)$ is the set of all concepts in the ontology i.e. $C_1(o) = \{Author, Person, Publication, JournalPaper, ConferencePaper\}$ and $C_2(o)$ is the set of some concepts, i.e. $C_2(o) = \{Author, Publication\}$ then $C_2(o) \subset_C C_1(o)$.

Definition 2 : Relation Subsumption ($\subset_R$) . A relation set R_1(g) is subset of another set R_2(g) if every edge (i.e., relation) in R_1(g) is in R_2(g). R_1(g) may have exactly the same edges (i.e., relations) as R_1(g).

$$R_1(\text{g}) \subset_R R_2(\text{g}) \text{ iff } \forall \text{ r, } \text{r} \in R_1(\text{g}) \rightarrow \text{r} \in R_2(\text{g}).$$

Considering Figure 8.1, if $R_1(o)$ is the set of some relations or edges in the ontology e.g., $R_1(o) = \{authorOf, publicationOf, title, journal, bookTitle\}$ and $R_2(o)$ is the set of object relations only i.e. $R_2(o) = \{authorOf, publicationOf\}$ then $R_2(o) \subset_R R_1(o)$.

Definition 3 : Ontology Pattern (op). An ontology pattern op in an ontology 'o' is a directed labelled graph, comprising of nodes and edges (i.e. concepts and relations), where

$C(op) \subset_C C(o)$ and $R(op) \subset_R R(o)$ and $\forall$ r(c_i, c_j), r(c_i, c_j) $\in$ R(op) $\rightarrow$ r(c_i, c_j) $\in$ R(o).

The example ontology in Figure 8.1 comprises of a set of concepts $C(o) = \{Author, ConferencePaper, JournalPaper, Person, Publication\}$ and a set of relations $R(o) = \{authorOf, bookTitle, journal, number, page, publicationOf, subClassOf, title, volume, year\}$. The part of ontology marked as red in Figure 8.1 (referred to as o_{red}) is composed of a set of concepts $C(o_{red}) = \{Author, ConferencePaper, Person, Publication\}$ and a set of relations $R(o_{red}) = \{authorOf, publicationOf, subClassOf\}$. Since $C(o_{red}) \subset_C C(o)$, $R(o_{red}) \subset_R R(o)$C and for all relations $r \in R(op)$ (i.e. $authorOf(Person, Publication)$, $publicationOf(Publication, Person)$, $subClassOf(Author, Person)$, $subClassOf(ConferencePaper, Publication)$) $r \in R(o)$, according to the definition 3, o_{red} is an op in o.

Definition 4 : Query Graph (QG) . Given a query string $Q = \{q_1, q_2, ..., q_n\}$ containing 'n' query terms, a *query graph* 'QG' w.r.t. Q is defined as an unlabelled undirected graph where each query term is mapped to a node, and each node has one or more unlabelled and undirected edges that connect this node to every other node in the graph. C(QG) is a set of nodes or concepts where $c_q \in C(QG)$ denotes a concept (query term) in 'QG'. R(QG) is the set of edges where (c_i, c_j) $\in$ R(QG) denotes a relationship between c_i and c_j.

An example three-term query string Q = 'author conference paper', and a possible query graph 'QG' for 'Q' is shown in Figure 8.2, where each query term $q \in Q$ is mapped to a query graph concept c_q and nodes are connected to each other.

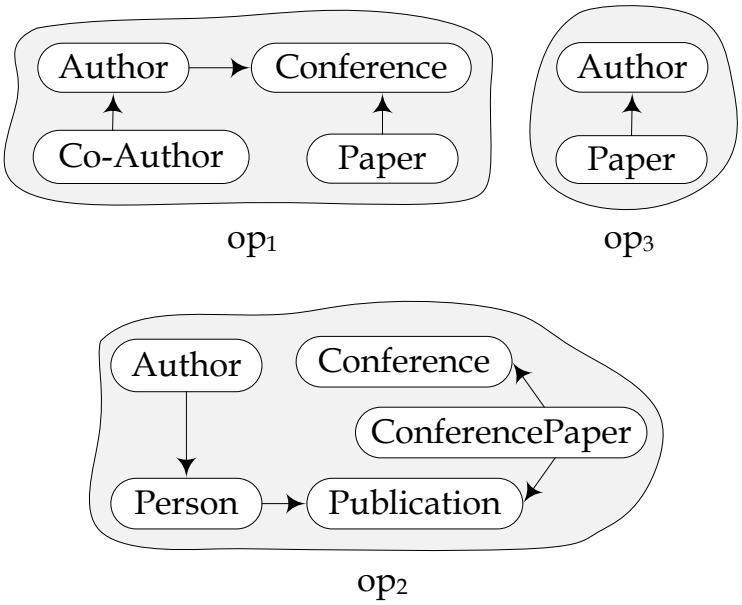


Figure 8.3: Match Patterns for 'Q'

Definition 5 : Query Match (Q_{mat}) . A query match Q^j_{mat} in an ontology 'o_j' is a set of ontology patterns 'op' in 'o' such that for a $C_i(QG)$ that is a subset of $C(QG)$, $C_i(QG)$ is also a subset of C(op).

$$Q_{mat} = \{\ op : C_i(QG) \subset_C \text{C(op) and } C_i(QG) \subset_C C(QG)\ \}$$

According to the definition, a query match is a partial match for a query graph. Consider an example query Q = 'author paper conference' , a query graph QG for this query consists of three nodes and therefore $C(QG)$= $\{Author, Paper, Conference\}$ as shown in Figure 8.2. Figure 8.3 shows example ontology patterns op_1, op_2 and op_3 in ontologies o_1, o_2 and o_3, respectively. Since, a subset of $C(QG)$ is also a subset of

each of $C(op_1)$, $C(op_2)$ and $C(op_3)$ therefore op_1, op_2 and op_3 are query matches o_1,o_2 and o_3 in for QG.

Due to the adaptation of a partial match approach in RecOn, a query match Q_{mat} may introduce some additional concepts or relations, or drop a few existing concepts and relations compared to the concepts and relations contained in the query graph QG itself as shown in matches 'op_1', 'op_2' and 'op_3' of Figure 8.3. The newly introduced concepts in Q_{mat} may or may not match a query node. For instance, the ontology pattern op_2 of Figure 8.3, introduces two new concepts 'Person' and 'Publication' that do not exist in the query graph. Similarly 'op_3' lacks the query concept 'conference'.

Definition 6 : Concept Match Set ($C_{Q_{mat}}$) . If a concept $c \in C(Q_{mat})$ i.e. concept set of query match in 'o', is a match for at least one of the query nodes $c_q \in C(QG)$ (i.e. $c \in C(Q_{mat}) \approx c_q \in C(QG)$), then the concept c is called a **match concept** $\phi(c_q)$ of c_q. The set of all matched nodes of a query graph match is the **concept match set** $C_{Q_{mat}}$ of the query graph match i.e.,

$$C_{Q_{mat}} = \{c : c \in C(Q_{mat}) \wedge c = \phi_i(c_q) \wedge c_q \in C(QG)\}$$

For instance, for a query 'q' and query graph 'QG', as shown in Figure 8.2, a query match Q_{mat} is 'op_2' (i.e. ontology pattern in 'o_2') as shown in Figure 8.3, the match for a query concept 'author', 'paper' and 'conference' is $\phi(author)$ = 'Author', $\phi(paper)$ = 'ConferencePaper', and $\phi(conference)$ = 'ConferencePaper' respectively. Therefore, the concept match set $C_{Q_{mat}}$ for Q_{mat} is {'Author', 'ConferencePaper'}.

8.2.2 RecOn Workflow

RecOn is implemented as a Java web application that uses Virtuoso as an ontology repository. Figure 8.4 shows the overall execution flow of RecOn. Starting from the input, four components participate in the ontology recommendation task. Here, a step-by-step explanation of how different RecOn components participate in the recommendation task is given.

8.2.2.1 Query preprocessing

This component takes a query string as an input and extracts keywords from the string by stemming and removing stop words. A query graph is generated from the extracted keywords, where each keyword is matched to a node and each node is connected to each other node in a query graph. If a keyword appears several times in the query string, only one corresponding node is created for that keyword in the query graph. A query graph considers each keyword as a single word; however, we look for them as compound words in their matches in the ontologies (cf. Section 8.3.2.1). The query graph is then used to find the appropriate ontology matches for the query.

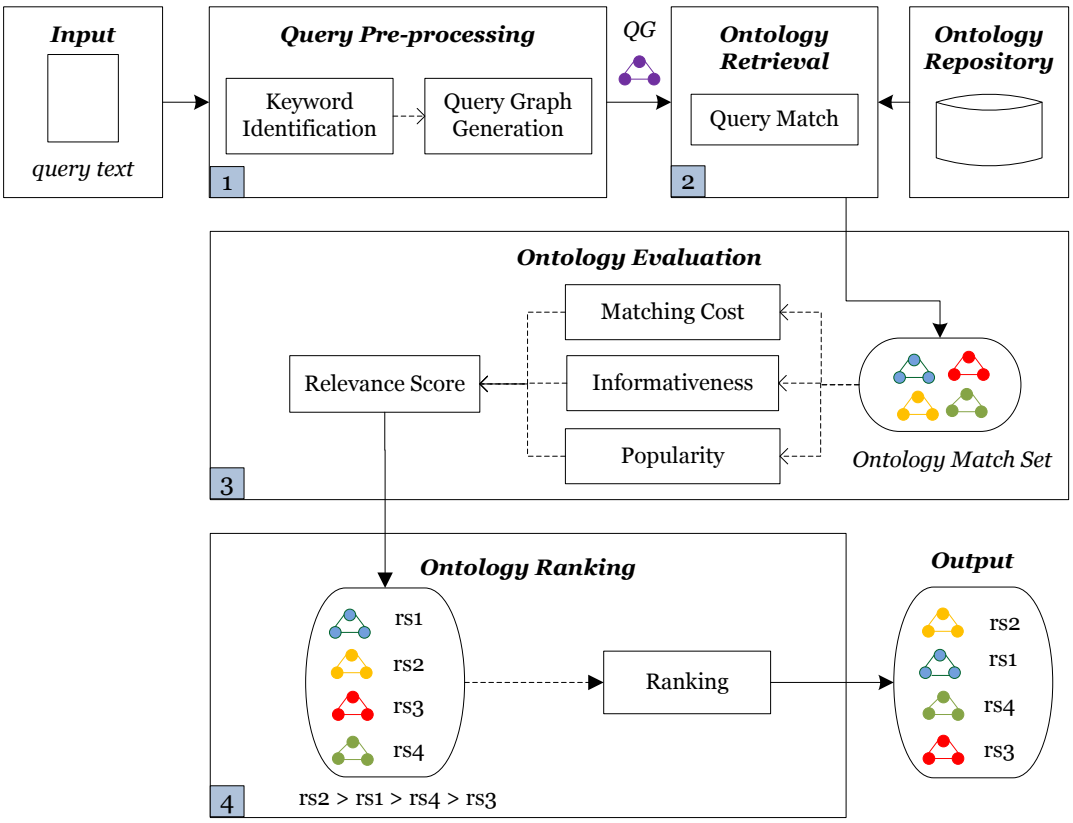


Figure 8.4: RecOn Workflow

8.2.2.2 Ontology Retrieval

This component considers a query graph and finds candidate ontologies for the query graph as discussed in Section 8.3.1. RecOn dynamically maps a query graph to a SPARQL query and retrieves the query matches. RecOn is implemented and tested for the English language only; therefore the SPARQL query is defined to look up English labels of concepts only (i.e. labels that are set with an *@en* tag). The output of this component is the ontology match set that is passed on to the next component for the ontology evaluation.

8.2.2.3 Ontology Evaluation

This component preprocesses query matches before evaluating the ontology match set. The labels from each query match are considered, and for each label, the language tag (i.e. , '*@en*') is removed. A labelSplit() function is used to split the label based on capital letters to retrieve all words from the label, and each word obtained from the label is then stemmed. For instance, '*ConferencePapers@en*' results, after removing the language tag, splitting the label and stemming in '*Conference Paper*'.

After this preprocessing the matching cost (cf. Section 8.3.2.1), informativeness (cf. Section 8.3.2.2), and popularity (cf. Section 8.3.2.3) for each ontology in the ontology match set are computed. Finally, the relevance score for each ontology to the user query is determined as shown in Section 8.3.2.4.

8.2.2.4 Ontology Ranking

This component orders the matching ontologies in order of their relevance score to the user query and outputs a ranked list of matched ontologies and their concepts.

8.3 Ontology Recommendation

Based on the functions and definitions above, we now explain our ontology recommendation model.
Given a query string Q and an ontology collection O, the purpose is to find O_{MAT} - a set of matching ontologies to the query string Q in O, and recommend up to k ontologies to the user to help her finding the right ontology. To achieve that, the ontologies that are relevant to the query are first retrieved and then k ontologies are selected based on their matching cost, informativeness and popularity to the query string.

8.3.1 Ontology Retrieval

Given an ontology collection O and a query string Q, to characterise the match of an ontology 'o' to Q, we define the candidacy of 'o' w.r.t. Q as

$$cand(o, Q) = \begin{cases} true & \text{if } \exists\ Q_{mat} \text{ in 'o'} \\ false & \text{otherwise} \end{cases} \tag{8.1}$$

which is either 'true' (a candidate ontology) or 'false' (not a candidate ontology). As mentioned in Equation 8.1, an ontology 'o' is a candidate ontology for a query Q, if it contains at least one match Q_{mat}.

Example: For instance, Figure 8.2 shows a query graph for a three keyword query string i.e., 'paper author conference'. The example ontology shown in Figure 8.1 contains a query match Q_{mat} (ontology pattern marked as red in Figure 8.1) for Q, therefore the example ontology is a candidate ontology for Q.

A set of all match ontologies in an ontology corpus O for a query Q is referred to as the **ontology match set** (O_{MAT}). i.e.,

$$O_{MAT} = \{o_i : cand(o_i, Q) \equiv true\} \tag{8.2}$$

8.3.2 Recommending k Ontologies

Among all the ontologies in the ontology, match sets O_{MAT} for a query Q, we aim to recommend up to k ontologies that are informative and popular while incurring the least matching costs. In the following, firstly we define the matching cost of an ontology to the query, the informativeness and the popularity of the ontology. Then we integrate these measures to get the final score for the ontologies.

8.3.2.1 Matching Cost

We consider the matching cost for each ontology $o_i \in O_{MAT}$ to find the k best matching ontologies out of the ontology match set. The matching cost for o_i is the difference between the content and structure of the concepts in the query match Q^i_{mat} and the corresponding query graph QG. To quantify the matching cost of an ontology o_i, the matching cost for a query match Q^i_{mat} is computed. The matching cost considers both, the structure and the content of a query match, referred to as the *structure matching cost* and the *label matching cost*, respectively.

Label matching cost. The label matching cost for a query match Q_{mat} is the average difference between the label of the matched concept c $\in C_{Q_{mat}}$ and its corresponding query concept $c_q \in C(QG)$, where c is a match of c_q i.e. $c = \phi(c_q)$, as shown in Equation 8.3 .

$$
\begin{aligned}
cost_{lb}(Q_{mat}) &= \frac{1}{|C_{Q_{mat}}|} \sum_{\forall c_q \in C(QG)} \sum_{\forall c \in C_{Q_{mat}}} dist_{lb}(c, c_q) : c = \phi(c_q) \\
dist_{lb}(c, c_q) &= \frac{|l_c \cup l_{c_q}| - |l_c \cap l_{c_q}|}{|l_c \cup l_{c_q}|}
\end{aligned} \tag{8.3}
$$

where, $dist_{lb}(c, q)$ is the difference in the labels' contents of c (i.e. l_c) and the query term c_q (i.e. l_{c_q}), and $|C_{Q_{mat}}|$ is the size of the concept match set. $dist_{lb}(c, c_q)$ is measured according to information retrieval principles and is computed by using the *Jaccard distance* metric. *Jaccard distance* is a commonly used measure of distance between two sets; we compute the *Jaccard distance* of the set of words. Here, l_c and l_{c_q} represent a set of words in the label of c and c_q respectively. $l_c \cup l_{c_q}$ is the set of all distinct words in labels of c and c_q, and $l_c \cap l_{c_q}$ are the common words in l_c and l_{c_q}.
`Example:` For a match 'op_2' in Figure 8.3, the $dist_{lb}(ConferencePaper, paper)$ is computed as:

$$
dist_{lb}(ConferencePaper, paper) = \frac{|\{conference, paper\} \cup \{paper\}| - |\{conference, paper\} \cap \{paper\}|}{|\{conference, paper\} \cup \{paper\}|} = \frac{2-1}{2} = 0.5
$$

Note that $dist_{lb}(c, c_q)$ is maximum (i.e. 1) if there is no match for a keyword in

this query match. The label matching cost is low for the query matches that contain all or more common words with the query concept labels. For example, in Figure 8.3, 'op_2' is favoured over match 'op_3' as a query match for QG, since 'op_2' contains more common words with concepts of QG than 'op_3'.

Structural matching cost. A structure matching cost measures the difference in the connectivity structure of the matched concepts of a query match Q_{mat}. The purpose of this metric is to prefer those ontologies that contain Q_{mat} where the concepts of concern are in close vicinity. The intuition behind this is, that the more the concepts are connected to each other, the more closely they are defined in the domain of discourse. An ideal match Q_{mat} is the one that is at least as connected as the least connected query graph QG, i.e. all the concepts in the match Q_{mat} should have a direct connection to at least one other query match in Q_{mat}. We compute the structural cost as:

$$
\begin{aligned}
cost_{st}(Q_{mat}) &= \frac{1}{n}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} dist_{st}(c_i, c_j) : c_i = \phi(c_{q_i}) \& c_j = \phi(c_{q_j}) \\
dist_{st}(c_i, c_j) &= \begin{cases} 0 & \text{if } c_i = c_j \\ |R_{SP}(c_i \rightarrow c_j)| & \text{if } c_i \neq c_j \\ +\infty & \text{if } \nexists c_i \vee \nexists c_j \end{cases} \\
n &= |C(QG)| \qquad (8.4)
\end{aligned}
$$

In Equation 8.4, $dist_{st}(c_i, c_j)$ is the minimum structural distance of any of the match for c_{q_i} and c_{q_j} i.e., concept c_i to c_j in Q_{mat}. The structural distance $dist_{st}(c_i, c_j)$ is 0 if $c_i = c_j$ (i.e., two query terms match to the same concept in an ontology are considered as a compound word in that ontology) or the shortest distance, in terms of number of edges, of concept c_i to c_j in Q_{mat}. The length of the shortest path between c_i and c_j is positive infinity (+∞) if c_i and c_j are disconnected (i.e., either one or both $\phi(c_{q_i})$ or $\phi(c_{q_j})$ does not exist in Q_{mat}. The structure cost of a query match $cost_{st}(Q_{mat})$ is the average distance among all the concepts of the concept match set $C_{Q_{mat}}$ of Q_{mat}.

`Example:` Let us consider two query concepts 'author' as c_{q_i} and 'paper' as c_{q_j} of QG, shown in Figure 8.2. The structural distance of the matches ($\phi(c_{q_i})$ and $\phi(c_{q_j})$) in op_1, op_2, and op_3, shown in Figure 8.3, is as follows:

op_1: $\phi(author)$ is 'Author' and $\phi(paper)$ is 'Paper', and $dist_{st}(Author, Paper)$ is 2.

op_2: $\phi(author)$ is 'Author' and $\phi(paper)$ is 'ConferencePaper', and $dist_{st}(Author, ConferencePaper)$ is 3.

op_3: $\phi(author)$ is 'Author' and $\phi(paper)$ is 'Paper', and $dist_{st}(Author, Paper)$ is 1.

To combine the structural cost and the label costs of the query fold match, we take

the harmonic mean of $cost_{lb}(Q_{mat})$ and $cost_{st}(Q_{mat})$ of Q_{mat}, as shown in Equation 8.5.

$$cost(Q_{mat}) = \frac{2.cost_{lb}(Q_{mat}).cost_{st}(Q_{mat})}{cost_{lb}(Q_{mat}) + cost_{st}(Q_{mat})} \quad (8.5)$$

The cost of an ontology o for Q is the cost of Q_{mat} in o as shown in Equation 8.6.

$$cost(o, Q) = cost(Q_{mat}) : Q_{mat} \in o \quad (8.6)$$

8.3.2.2 Informativeness

Informativeness is defined as a measure of knowledge an ontology provides about a user query. The informativeness measure is characterized by a feature that an ontology 'o' is more informative for a query Q, if a query match Q_{mat} exists in 'o', and the concepts in $C_{Q_{mat}}$ have more relations with other concepts of that ontology or they have datatype relations defined for them. The intuition behind this is, the more relations that are defined for a concept, the more important the concept is in the ontology, and the more information it includes. Therefore, an ontology in which the concepts defined in the ontology have no or very few relations with other concepts in the ontology are considered less informative for a query.

Our algorithm prefers to recommend more informative ontologies, i.e. ontologies that have stronger connections between its concepts/datavalues. More precisely, the informativeness of an ontology 'o' is a measure regarding the informativeness of the query match Q_{mat} it contains. To quantify the informativeness of a query match Q_{mat}, we measure the informativeness of each concept in $C_{Q_{mat}}$.

The informativeness of a concept of $C_{Q_{mat}}$ is quantified by measuring the connectivity of the concept in its ontology. The informativeness of each concept c $\in C_{Q_{mat}}$ is the informativeness of the event that 'c' is indeed observed as a concept involving relations in an ontology.

$$inf(c, C_{Q_{mat}}) = 1 + log\frac{rf(c,o)}{max\{rf(c_j, o_j) : c_j \in C_{Q^j_{mat}} \bigcap c \wedge c_j = \phi(c_q) : c_q \in QG\}}$$

$$rf(c,o) = |\{r \in R(o) : domain(r) \vee range(r) = c \; or \; super(c)\}| \quad (8.7)$$

$Inf(c, C_{Q_{mat}})$ of a concept is a query dependent metric as shown in Equation 8.7. The informativeness of c of a query match $C_{Q_{mat}}$ is equal to the log of $rf(c,o)$, the *relation frequency* of the concept in the ontology it belongs to, divided by the maximum $rf(c_j, o_j)$ of the concept c_j in its home ontology o_j, where c_j belongs to a query match Q_{mat} for the query Q in o_j, and c and c_j are concept matches for the same query node $c_q \in QG$ in Q_{mat} in and Q^j_{mat}, respectively.

Example: Let us suppose that op_1, op_2, and op_3 (resp. o_1, o_2, and o_3) are the

only query matches (resp. matched ontologies) for the example QG in an ontology corpus. For $Q_{mat} = op_3$, C_{op_3} = {Author, Paper}, and

$$\begin{aligned} inf(Author, C_{op_3}) &= 1 + log\frac{rf(Author, o_3)}{rf(Author, o_1)} = 1 + log\frac{1}{2} = 0.7 \\ inf(Paper, C_{op_3}) &= 1 + log\frac{rf(Paper, o_3)}{rf(ConferencePaper, o_2)} = 1 + log\frac{1}{2} = 0.7 \end{aligned}$$

Based on the informativeness of the concepts of a query match, we compute the informativeness of the query match Q_{mat} as shown in Equation 8.8.

$$inf(Q_{mat}, o) = \frac{1}{|C(QG)|} \sum_{\forall c: c \in C_{Q_{mat}}} inf(c, C_{Q_{mat}}) \tag{8.8}$$

Example: For the example query match op_3,

$$\begin{aligned} inf(op_3, o) &= \frac{1}{|\{Author, Paper, Conference\}|}(inf(Author, C_{op_3}) + inf(Paper, C_{op_3}) \\ &= \frac{1}{3}(0.7 + 0.7) = 0.47 \end{aligned}$$

The informativeness of an ontology is then measured as the informativeness of a query match it contains for a query Q as shown in Figure 8.6.

$$Inf(o, Q) = inf(Q_{mat}, o) \tag{8.9}$$

8.3.2.3 Popularity

The *popularity* of an ontology is measured based on the level of reuse of the ontology in an ontology corpus or based on the size of RDF data populated according to the ontology. In this chapter, the *popularity* of an ontology is measured regarding its reuse within the ontology corpus. Therefore, we define *pop(o,O)*, to measure the *popularity* of an ontology o in ontology corpus O. Our *popularity* function is characterised by the following two features: (i) *reuse:* an ontology is more popular, if there are more ontologies using the ontology. (ii) *neighbourhood:* an ontology is more popular, if other popular ontologies use the ontology. Based on these two features, a *reuse* of ontology o by ontology o_i is considered as a "positive vote" for the popularity of ontology o from o_i. The PageRank algorithm is adopted as the *popularity function*, whereby each ontology is considered a node. Equation 8.10, formalises the *popularity function* which computes the popularity of o at the *kth* iteration.

$$pop_k(o,O) = \frac{1-\alpha}{|O|} + \alpha \sum_{o_i \in BO(o)} \frac{pop_{k-1}(o_i,O)}{|FO(o_i)|} \quad (8.10)$$

In Equation 8.10, |O| is the total number of ontologies in the ontology corpus, $BO(o)$ is a set of ontologies reusing the ontology o and $FO(o)$ is a set of ontologies, ontology 'o' is reusing. We further normalised the popularity of an ontology within the matched ontology set for query, i.e., O_{MAT}.

$$pop(o,Q) = \frac{pop(o,O)}{max\{pop(o_j,O) : o_j \in O_{MAT}\}} \quad (8.11)$$

In Equation 8.11, $pop(o,Q)$ returns a value from [0-1]. It is the relative popularity of an ontology 'o' in O_{MAT} that is achieved by dividing the popularity of o with the maximum popularity for any ontology among the matched ontologies for query Q.

8.3.2.4 Relevance Score

Finally, we define the relevance score of an ontology 'o' to the query Q, as a function of the matching cost, the informativeness and the popularity of 'o' for Q.

Linear Model: We describe a linear model containing fixed weights as a quantitative metric to measure the overall relevance between the query Q and the ontology 'o', and choose the up to k ontologies that have high relevance to the query.

$$rel(o,Q) = \alpha\ [inf(o,Q)] + \beta\ [pop(o,Q)] + \gamma\ [\frac{1}{cost(o,Q)}] \quad (8.12)$$

According to Equation 8.12, ontologies with high informativeness and popularity, and low matching costs are preferred among all matching ontologies O_{MAT}. Here, α, β and γ are the variable sets to combine the three features of a linear model.

Optimisation Problem: In O_{MAT}, the set of ontologies matched to Q in O, we aim to find up to k ontologies that are as informative and popular as possible while having the least matching costs (i.e. the best matches to the query). It can be formulated as an optimisation problem, in particular, as a 2-dimensional 0-1 knapsack problem, where each $o_i \in O_{MAT}$ corresponds to an 'item' to be selected whose 'value' v_i is the informativeness and popularity, and 'weight' w_i is 1, when the 'capacity' of the 'knapsack' w.r.t the number of items is k and w.r.t. the matching cost is γ .

$$
\begin{aligned}
&\textit{maximize} \sum_{\forall i: o_i \in O_{MAT}} x_i * [\alpha(inf(o,Q)) + \beta(pop(o,Q))] \\
&\textit{subject to} \\
&\sum_{\forall i: o_i \in O_{MAT}} x_i \leq k \\
&\textit{minimize} \sum_{\forall i: o_i \in O_{MAT}} x_i * cost(o_i, Q) \\
&x_i \in [0,1], 1 < i < |O_{MAT}|
\end{aligned} \tag{8.13}
$$

In Equation 8.13, the optimisation algorithm maximises the informativeness and popularity of the ontology in the result set, where α and β are the variable sets to combine the two, considering the constraint of the result set size, i.e. k and its matching costs.

We implemented the 2-dimensional knapsack problem as a less optimum greedy algorithm solution. An optimal solution using dynamic programming takes a lot of time but results in an optimal solution, whereas a less optimum greedy solution is efficient, but the results are not optimal. The greedy algorithm first sorts the ontologies in increasing order of their matching cost and then selects the one with high popularity and informativeness. If two of the ontologies have the same matching cost, it prefers the one that is first evaluated for its popularity and informativeness.

8.4 Evaluation

In this section, we report on a set of experiments and a user study that we performed to demonstrate the effectiveness and efficiency of *RecOn*.

8.4.1 Experimental Setup

For our experiments, we use our previously established CBRBench [Butt et al., 2014a]. It contains an ontology collection, benchmark queries, and a gold standard for the benchmark queries as shown in Chapter 5. All the experiments are performed on a machine with Intel Core i7 3.4 GHz Octa-core CPU and 8GB RAM.

8.4.1.1 DataSet

For our ontology corpus we use the CBRBench ontology collection. This ontology collection is composed of 1011 OWL & RDF(S) ontologies that we use as our ontology corpus. We stored each ontology as a named graph in a Virtuoso database.

8.4.1.2 Query selection

CBRBench contains ten single-term queries and a gold standard composed of a relevance score for matching concepts to the queries on the task of ontology concept retrieval. CBRBench queries are selected using the query log[2] of the Linked Open Vocabularies (LOV) search engine [Vandenbussche and Vatant, 2014]. The most popular search terms in the log covering the period between 06/01/2012 and 16/04/2014 are selected as benchmark queries. These general queries cover a wide range of domains from the list of the most popular query terms in the LOV query log. This helped evaluators who were experts in different domains to correctly evaluate the concepts. All CBRBench queries are single word queries – that is for two reasons. First, only about 11% of all queries posed on the LOV search engine use compound search queries and no compound query was among the 200 most used queries and second, for no compound query in the top 1000 query terms did the benchmark collection contain enough relevant resources to arrive at a meaningful ranking. However, varying length queries composed of multi-terms are required to evaluate the effectiveness of *RecOn* that are not available in CBRBench. Thus, we first need to establish a set of queries to be used in our experiment as well as in future research. Our queries are derived from our earlier established CBRBench queries and ontology collection in two ways:

Single-term queries. Single-term queries proposed in the CBRBench are used as is to evaluate the performance of the ranking algorithms on ontology ranking. For each single-term benchmark query, the relevance score of the matched concepts to the query terms in the gold standard is considered the relevance of the corresponding ontologies to the query term. The ten single keyword queries used for the evaluation of *RecOn* are: `'address'`, `'author'`, `'event'`, `'location'`, `'music'`, `'name'`, `'organization'`, `'person'`, `'time'` and `'title'`.

Multi-term queries. Multi-term queries are created to evaluate the effect of the query size on the performance of the algorithm as follows.

1. First, for each of the ten query terms of CBRBench the top three matching ontologies are considered. This results in a collection of 28 ontologies (some ontologies appear in the top three for more than one query) while for two queries we considered four ontologies because there was a tie for the third-ranked ontology.

2. Each concept in these ontologies is assigned a single-term label by finding the *intended type of class* using the method described in [Butt et al., 2014b], e.g., the label "An Organization - a base class for instances of organizations" for a class[3] is reduced to 'organization'.

[2]See http://lov.okfn.org/dataset/lov/stats/searchLog.csv
[3]http://data.press.net/ontology/stuff/Organization

3. Once each concept has a single-term label, all possible combinations of length 2, 3 and 4 terms are generated for each ontology from the labels of the concepts of the ontology. E.g., a string 'person & university & student' is generated by combining the labels of the 'person', 'university' and 'student' classes for the query term `'person'` from one of its matched ontology i.e., UNIV_BENCH ontology[4].

4. For each combination of concept labels the number of times they occur collectively in the ontology corpus is computed. The most frequently occurring 2, 3 and 4 length concept label combinations are then selected from each ontology. The intuition behind this process is that the more often concepts occur together in the ontology corpus, the more related they are (they belong to the same domain of discourse). We could not find a meaningful combination for some multi-term queries because they do not occur together in any other ontology and ended up with 30 additional multi-term queries to evaluate `RecOn` as shown in Table 8.3.

8.4.1.3 Baseline:

To evaluate the quality of results produced by *RecOn*, two versions of *RecOn* are implemented.

- $RecOn_{opt}$ - Optimized RecOn , *RecOn* recommends up to 'k' results based on the relevance score computed through the optimisation model. The weights for calculating relevance score (Equation 8.12) for our experiments are set to 0.4, 0.3, and 0.3 for the 'matching cost', 'informativeness' and 'popularity' metrics respectively. The relative weighs for these metrics are selected based on how well each metrics performed in our pre-evaluation tests.

- $RecOn_{ln}$ - Linear RecOn , *RecOn* recommends up to 'k' results based on relevance score computed with linear relevance model.

To evaluate the effectiveness, we compare the result set of $RecOn_{opt}$ with the result sets of `AKTiveRank` and $RecOn_{ln}$ for all queries shown in Table 8.3.

[4] http://swat.cse.lehigh.edu/onto/univ-bench.owl#

Table 8.3: Query strings derived from benchmark query terms

Q-Id	Multi term queries	Q-Id	Multi term queries
Q1	`person` & agent	Q20	`name` & person
Q2	`person` & organization	Q21	`name` & title
Q3	`person` & organization & project	Q22	`name` & person & agent
Q4	`person` & student & professor & university		
Q5	`organization` & location	Q23	`title` & identifier
Q6	`organization` & student	Q24	`title` & organization
Q7	`organization` & student & course	Q25	`title` & author & document
Q8	`organization` & student & course & university		
Q9	`event` & location	Q26	`location` & place
Q10	`event` & conference	Q27	`location` & place & geographic
Q11	`event` & conference & paper		
Q12	`event` & conference & paper & article		
Q13	`author` & publication	Q28	`music` & event
Q14	`author` & newspaper	Q29	`music` & group
Q15	`author` & publication & research		
Q16	`author` & publication & research & issue		
Q17	`address` & organization	Q30	`time` & date
Q18	`address` & organization & place		
Q19	`address` & organization & place & country		

AKTiveRank is considered our baseline, since the approach is one of the two state-of-the-art generic ontology ranking techniques (the second one is Swoogle [Ding et al., 2005]). However, as the evaluation results presented in [Alani et al., 2006] prove, AKTiveRank outperforms Swoogle on the task of ontology ranking. Another reason for not considering Swoogle as a baseline is that it computes the ranks for the matched ontologies by the instances of that ontology in the Swoogle database, which is not possible in our case where the dataset is merely composed of ontologies. A list of $RecOn_{opt}$ containing 'k' elements is compared with $RecOn_{ln}$ to verify that the quality of the results produced by the optimisation model is better than a linear model.

8.4.2 User Study

8.4.2.1 Approach.

We implemented $RecOn_{opt}$ and $RecOn_{ln}$, and re-implemented *AKTiveRank* to the best of our abilities. A list of relevant results for all three models is produced for multi-term queries over the CBRBench dataset to compare the effectiveness of *RecOn* in a user study. We conducted the user study with sixteen human experts from the ANU, Monash University, the University of Queensland, CSIRO, Fraunhofer Institute, Vienna University of Business, KIT, Universidad de Chile, the Australian Bureau of Statistics and the Polytechnical University of Madrid. All of the evaluators have developed ontologies before, and some are authors of widely cited ontologies.

Table 8.4: Comparison statistics of $RecOn_{opt}$ with baselines

Baseline	Minimum	Maximum	Average	SDev
AKTiveRank	3	12	8	1.95
$RecOn_{ln}$	4	13	8	1.92

For the user study, an evaluation tool[5] was developed that allowed the experts to evaluate $RecOn_{opt}$ for the thirty multi-term query strings in comparison to the *AKTiveRank* and $RecOn_{ln}$ produced result sets. To make our evaluation robust and neutral, the following decisions were taken: (1) The order of the queries along with their relevant results shown to the evaluators was chosen randomly. Every participant was shown queries in a random order to eliminate the effect of the query sequence on the performance of the approaches. (2) For each query, two lists 'List A' and 'List B', each list consisting of the ten most relevant ontologies along with the matched concepts in the ontology, were shown to the participants. For each query $RecOn_{opt}$ was compared with one of the other two algorithms; either *AKTiveRank* or $RecOn_{ln}$ as the baseline. The selection of the baseline was random. (3) The positioning of the ranked lists as 'List A' or 'List B' was random too, i.e. results of $RecOn_{opt}$ appeared either as 'List A' or 'List B'. Table 8.4 shows the statistics about the evaluation strategy. The Table shows minimum, maximum, average, and standard

[5] http://activeraul.org/ontologySearch/

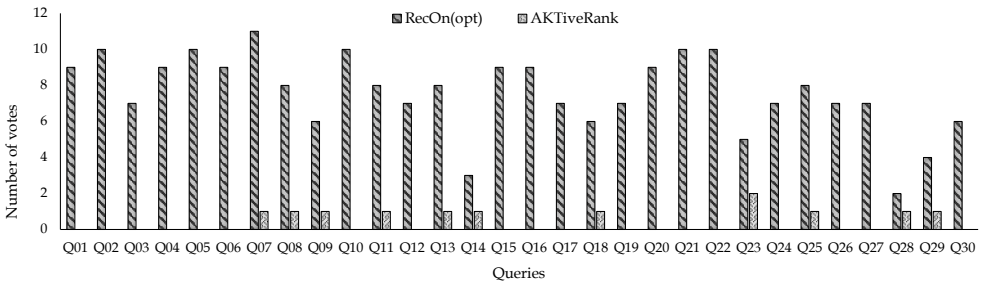


Figure 8.5: Effectiveness of $RecOn_{opt}$ vs. *AKTiveRank* for multi-term query strings

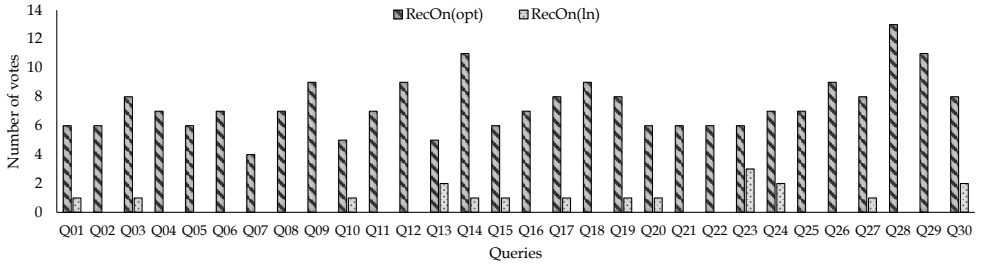


Figure 8.6: Effectiveness of $RecOn_{opt}$ vs. $RecOn_{ln}$ for multi-term query strings

deviation of the number of times the results of $RecOn_{opt}$ for a query was evaluated in comparison to the *AKTiveRank* or $RecOn_{ln}$ results. The statistics show a balance in the evaluation as on average eight evaluations for each query were received for $RecOn_{opt}$ in comparison to both baselines.

8.4.2.2 Results

To derive our results, we considered a positive vote for $RecOn_{opt}$ (resp. $RecOn_{ln}$ or *AKTiveRank*) when the list comprised of results of $RecOn_{opt}$ (resp. $RecOn_{ln}$ or *AKTiveRank*) was selected by the expert as the "more relevant" result set for a given query. Figure 8.5 and Figure 8.6 show all 16 votes for all thirty queries; the x-axis shows all 30 queries while the y-axis shows the number of votes (evaluations) in favour of each approach in comparison to the one other approach. $RecOn_{opt}$ (resp. *AKTiveRank*) bars in Figure 8.5 show the number of votes for *OptimizedRecOn* (resp. *AKTiveRank*) in comparison to *AKTiveRank* (resp. *OptimizedRecOn*). Similarly, the results for $RecOn_{opt}$ and $RecOn_{ln}$ in Figure 8.6 show the number of votes for *OptimizedRecOn* in comparison to *AKTiveRank*. Figure 8.5 shows that $RecOn_{opt}$ incurred more positive votes in comparison to the baseline *AKTiveRank*, i.e. 94% of the time experts voted for the result set produced by $RecOn_{opt}$ to be "more relevant" than `AktiveRank` for the example queries. Moreover, 92% of the time $RecOn_{opt}$ generated "more relevant" result-sets in comparison to $RecOn_{ln}$ as shown in Figure 8.6.

Figure 8.7 shows the effect of the query size on the effectiveness of $RecOn_{opt}$. The

percentage of positive votes, in comparison to the baselines, for 2, 3, and 4 terms queries are shown here. The figure confirms that an increase in the number of the query terms increased the effectiveness of $RecOn_{opt}$ in comparison to the baselines. The average of positive votes increased from 90% to 98% (resp. 92% to 98%) and the standard deviation decreased from 11.9% to 4.9% (resp. 11.2% to 4.9%) in comparison to *AKTiveRank* (resp. $RecOn_{ln}$).

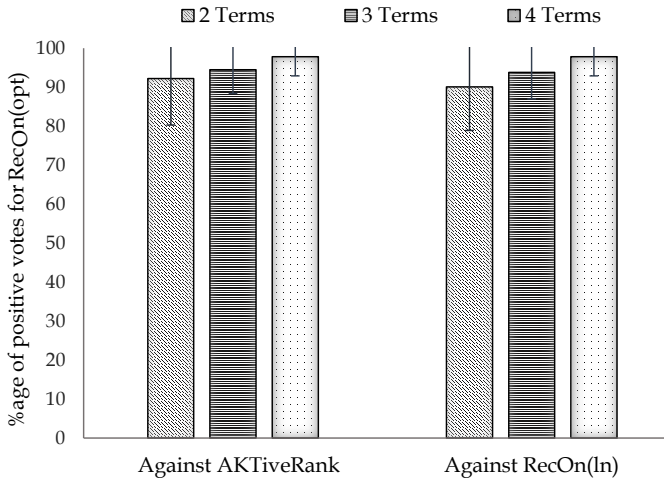
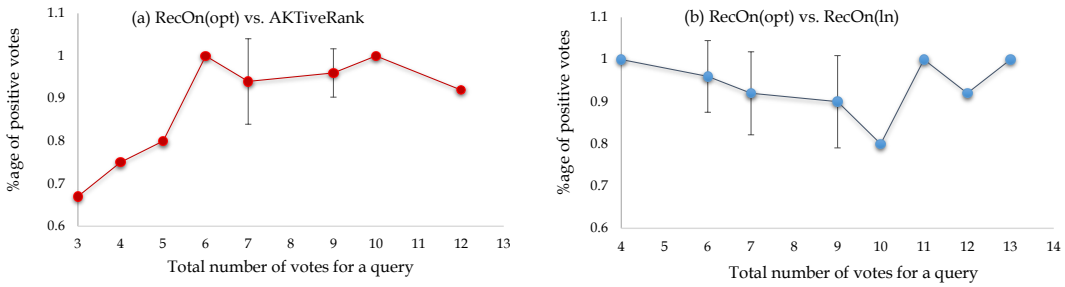


Figure 8.7: Effectiveness of query length on the $RecOn_{opt}$ performance



(a) $RecOn_{opt}$ against *AKTiveRank* (b) $RecOn_{opt}$ against $RecOn_{ln}$

Figure 8.8: Effect of number of evaluations (votes) on $RecOn_{opt}$ performance

In another analysis, we examined the effect of the number of evaluations for a query on the performance of $RecOn_{opt}$. Figure 8.8, shows the statistics, the x-axis shows the number of evaluations (votes) for a query and y-axis shows the performance of $RecOn_{opt}$ in comparison to both baselines. The results presented here show that an increase in the number of evaluations corresponding to a query resulted in a stable and improved performance of $RecOn_{opt}$.

8.4.3 Experiments

Other than the user study, we also compared our approach to the gold standard available as part of the CBRBench benchmark. As mentioned in Section 8.4.1 we consider the rank of a resource in CBRBench, for a given query, as the rank for the ontology this resource belongs to. If more than one matched resources for a query term belong to the same ontology, the highest rank of a matched resource that belongs to the same ontology is assigned as the rank of the ontology for a given query term. We then measure the Average Precision (AP@10) and Normalised Discounted Commulative Gain NDCG@10 for all ten single term queries based on the gold standard derived from the CBRBench gold standard.

Table 8.5: AP and NDCG for Single term queries

Metric	Approach	Queries									
		Person	Name	Event	Title	Loc.	Addr.	Music	Org.	Author	Time
AP@10	$RecOn_{opt}$	**0.87**	**0.69**	**0.74**	**0.62**	**0.53**	**0.72**	**0.64**	**0.88**	**0.65**	0.41
	$RecOn_{ln}$	0.61	0.33	0.45	0.31	0.42	0.55	0.39	0.61	0.41	**0.46**
	AKTiveRank	0.47	0.16	0.25	0.34	0.34	0.48	0.42	0.32	0.54	0.23
NDCG@10	$RecOn_{opt}$	**0.66**	**0.51**	**0.49**	**0.37**	**0.32**	**0.54**	**0.42**	**0.54**	**0.43**	**0.28**
	$RecOn_{ln}$	0.43	0.21	0.29	0.33	0.27	0.38	0.33	0.39	0.25	0.19
	AKTiveRank	0.32	0.11	0.14	0.20	0.16	0.23	0.33	0.18	0.29	0.09

Table 8.5 shows the AP@10 and NDCG@10 of $RecOn_{opt}$, $RecOn_{ln}$ and *AKTiveRank* on ten single term queries. The results shows that $RecOn_{opt}$ performs better than $RecOn_{ln}$ and *AKTiveRank*. Moreover, $RecOn_{ln}$ outperforms *AKTiveRank* on all ten

sample queries.

For $RecOn_{opt}$, the Mean Average Precision (MAP@10) is 0.68, and average NDCG@10 is 0.46 that is better than the two baselines. The MAP@10 and NDCG@10 are 0.45 and 0.31 respectively for $RecOn_{ln}$, and MAP@10 and NDCG@10 are 0.36 and 0.21 for *AKTiveRank*.

8.4.4 Scalability Analysis

In the final experiment, we demonstrate the scalability of our approach and runtime improvements of $RecOn_{opt}$ over $RecOn_{ln}$. $RecOn_{opt}$ employs a greedy algorithm solution that tries to minimise the matching cost, to quickly find high-quality matches. We choose k = 10 and use the same queries as in the previous experiments (both single term and multi-term queries). The analysis is conducted on four ontology corpora with differing sizes in terms of the number of ontologies it contains (i.e. 10N, 100N, 1000N and 10000N).

The following experiments are based on varying sized corpora that are randomly sampled from the CBRBench. The ontologies in CBRBench varies in terms of the number of triples they contain. Therefore, a random selection of ontologies for all four corpora has high chances of measurement bias. To minimise the bias, we randomly generated ten samples of each corpus size and recorded the query runtime for all 40 queries on each sample. An average of 40 queries on ten samples is recorded as query execution time for one ontology corpus.

Table 8.6: Comparison statistics of $RecOn_{opt}$ with baselines

Ontologies Count	Minimum Triples	Maximum Triples	Average Triples	Standard Deviation
10	2357	103548	38782.5	41504.4
100	215934	765842	467776.1	174549.5
1000	3021562	8453289	5798053.9	1604409
10000	35381037	68672132	53217881.8	10667258.4

Table 8.6 shows the maximum, minimum and average number of triples for the samples of all 10N, 100N, 1,000N and 10,0000N corpora. The statistics show that samples for each size of ontology corpus vary in the number of triples. A measurement on a single sample may not be the true reflection of the performance. Therefore, for all experiments reported in this section, we consider an average of ten samples for an ontology corpus.

We recorded the query execution time for all single-term and multi-terms queries. The experiments are divided into those that measure the query match time i.e., the time it takes to find matched ontologies (as discussed in Section 8.3.1), and those that measure the top-k recommendation time, i.e. the time it takes to recommend up to 'k' high-quality matches for a query. The former are referred to as '`Query match time`' experiments and the latter as '`Recommendation time`' experiments. The reasons for this is two folds: 1) query match time is dependent on the under-

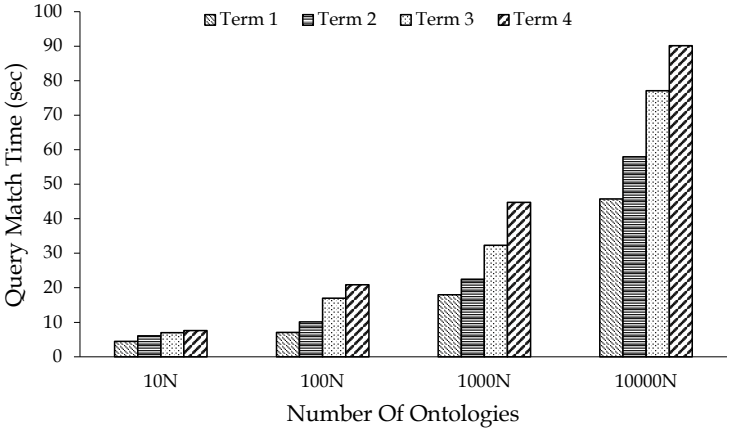


Figure 8.9: Query match time for different length queries

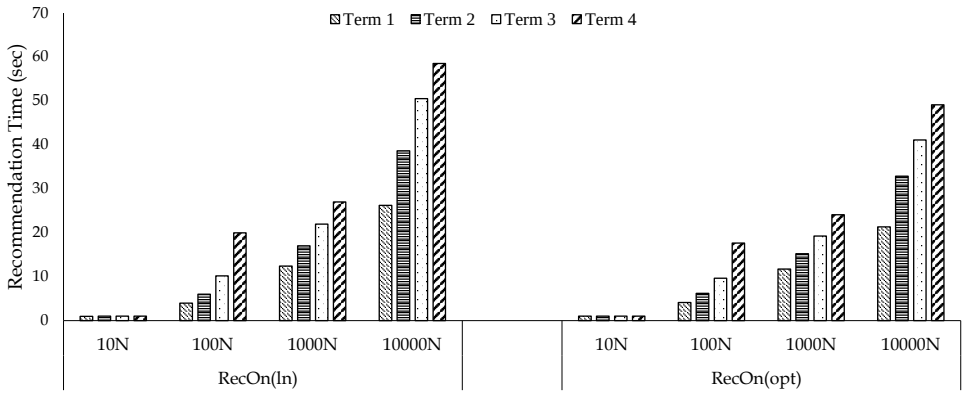
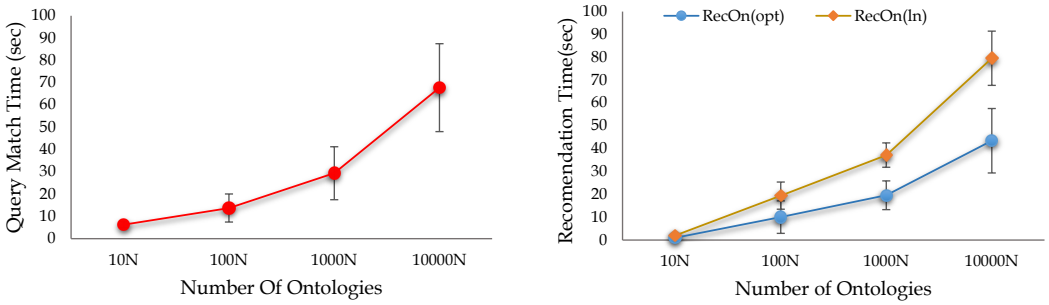


Figure 8.10: Query recommendation time for different length queries

lying RDF store (Virtuoso repository in our implementation), while for 2) for both $RecOn_{opt}$ and $RecOn_{ln}$ the query match time is the same, but the time to recommend k ontologies differs, due to the different implementations of the recommendation model.

The query match time here also contains the connection time with a local Virtuoso repository, which is the same for each query on all different sized ontology corpora. Figure 8.9 shows the query match time in seconds with varying query size, and varying corpus size, and Fig 8.11(a) presents an average query match time for varying corpus sizes. The figures show an increase in the query match time with an increase in the number of ontologies (resp. triples) and query size (i.e., the number of terms in a keyword query). The increase in the query match time is logarithmic with the increase in the number of ontologies/triples in *RecOn*. However, it performs sufficiently fast for a reasonable size of ontologies.

Similarly, Figure 8.10 and 8.11(b) show the ontology recommendation time in sec-



(a) Average query match time

(b) Average query recommendation time

Figure 8.11: Average query execution time for $RecOn_{opt}$ and $RecOn_{ln}$

onds with varying query size, and varying corpus size, for $RecOn_{opt}$ and $RecOn_{ln}$. The results show that $RecOn_{opt}$ performs better than $RecOn_{ln}$. The reason for improved performance is that we implemented a greedy algorithm for the knapsack optimisation that measures the matching cost for all the matched ontologies while recording the popularity and informativeness only until the top-k ontologies are retrieved. Although $RecOn_{opt}$ is more efficient in our experiments, the runtime for *RecOn* can further be reduced by employing a multi-threaded implementation.

8.4.5 Results Summary

From the statistics presented in Section 8.4.2 and Section 8.4.3, it is evident that $RecOn_{opt}$ outperforms the state-of-the-art ontology ranking algorithm *AKTiveRank* on the sample queries for the CBRBench ontology collection; while an optimized solution to recommend relevant ontologies is preferred over a linear model of *RecOn* in most cases. On average $RecOn_{opt}$ received 94% positive votes as compared to *AKTiveRank* and 92% positive votes as compared to $RecOn_{ln}$ of relevant ontologies for the multi-term queries in the user study. The results also show that the effectiveness of $RecOn_{opt}$ increases with an increase in the length of the query string (i.e., number of query terms); and an increase in the number of votes for a query result set increases the average of positive votes for $RecOn_{opt}$ which means that the lowest positive votes $RecOn_{opt}$ received for a query (i.e. 67%) could have improved with more evaluations for this query. However, in this user study, for multi-term queries the relevance and the order were evaluated at once by performing a comparative study of two lists. As future work, we aim to test them separately to improve the ranking independently of the relevance of an ontology.

Similarly, for a single term query strings, $RecOn_{opt}$ scored higher for MAP@10 (i.e. 0.68) when compared to *AKTiveRank* (i.e. 0.36) and higher for NDCG@10 (i.e. 0.46) when compared to the *AKTiveRank* (i.e. 0.21). This statistic shows that even for the evaluation of the order of a ranking list, $RecOn_{ln}$ yields better scores as compared to *AKTiveRank*. Further experiments were conducted to evaluate the design decisions

made for the proposed recommendation model (i.e. *RecOn*). The results presented in Section 8.4.4 show that $RecOn_{opt}$ performs better than $RecOn_{ln}$ in terms of total query execution time because of the greedy implementation of the knapsack optimisation.

8.5 Summary

In this chapter we have presented *RecOn*, an ontology recommendation approach to select and rank relevant ontologies against a multi-term structureless query. Our approach first finds a set of matched ontologies for a query string and then identifies the up to k most relevant matches using three measures, the *Matching cost*, the *Informativeness* and the *Popularity* of the matched ontologies. Then we integrate these measures by formulating and solving a linear model (i.e. $RecOn_{ln}$) and then as an optimisation problem (i.e. $RecOn_{opt}$). The evaluation of our approach against *AKTiveRank* and a linear version of our algorithm $RecOn_{ln}$ in a user study performed on the CBRBench dataset [Butt et al., 2014a] shows that $RecOn_{opt}$ outperforms the baseline state-of-the-art algorithm *AktiveRank* in 94% of the cases and $RecOn_{ln}$ in 92% of the cases. Moreover, our experiments also show that $RecOn_{opt}$ is efficient in terms of query execution time on the CBRBench ontology collection for the sample queries.

Chapter 9

Conclusion and Future Work

In this thesis, we described the Semantic Web data retrieval process in general, identified the problems of retrieval techniques, and proposed various solutions that address some of the challenges related to such techniques. In this chapter, we conclude the work presented in this thesis and provide future research directions. We summarise our contribution in Section 9.1 and discuss directions for future research in Section 9.2.

9.1 Summary of our Contributions

The contributions of our work to achieve the research objectives of this thesis are listed below:

1. **A Taxonomy of Semantic Web data Retrieval Techniques:** To address the RQ1, i.e., the challenge of identifying the limitations of existing approaches for ontologies retrieval, we proposed a taxonomy of Semantic Web data retrieval techniques in Chapter 4. Based on our proposed taxonomy we characterised existing techniques that have been developed for Semantic Web data retrieval in the last two decades (existing techniques are reviewed in Chapter 3). The taxonomy contains five main topics, namely *retrieval aspects*, *storage and search*, *ranking*, *evaluation*, and *practical aspects*, each of which contains three to four dimensions (resulting in a total of sixteen dimensions). The characterization of existing techniques along these sixteen dimensions of our taxonomy (in Table 4.4 on page 73) allowed us to identify gaps in existing approaches and research directions. We formulated research questions along the five main topics of our taxonomy based on this study and addressed some of the identified gaps in this thesis.

2. **An Evaluation Benchmark for Ontology Ranking Techniques:** In Chapter 5, we address two of the research questions: **RQ6** i.e., how to evaluate the newly emerging ontology libraries and search engines in comparison to existing ones? and **RQ2** i.e, do the document ranking models suffice for ontology ranking? In this regard, we designed a benchmark suite named CBRBench, for Canberra Ontology Ranking Benchmark, including an ontology collection, a

set of queries and a ground truth established by human experts for evaluating ontology ranking algorithms. We also presented a methodology for resource ranking evaluation where we discussed many of the decisions that need to be made when designing a search evaluation framework for resources defined in ontologies. Moreover, the evaluation of eight ranking algorithms through these benchmarks is conducted. Finally, we identify that document ranking models do not suffice for ontology ranking, highlight the performance limitations of existing ranking models, and present a set of recommendations derived from an analysis of our experiment that we believe can significantly improve the performance of the ranking models.

3. **DWRank - *Dual Walk Ranking Model for Concept Search:*** The contribution addresses the research question **RQ3** i.e., how to rank relevant resources and ontologies for keyword queries? We proposed DWRank a two-staged bi-directional graph walk ranking algorithm to rank concepts in ontologies based on how well they present a given search term in Section 6.3.1. The aim of our technique is to provide a ranking model that is suitable to rank concepts for a user keyword query on a domain independent ontology collection. We implemented and applied this algorithm on the task of searching and ranking concepts in ontologies and compare it with state-of-the-art ontology ranking models and traditional information retrieval algorithms such as PageRank and tf-idf. Our evaluation shows that DWRank significantly outperforms the state-of-the-art ranking models on the task of ranking concepts in ontologies for all benchmark queries in the ontology collection.

4. **Concept Retrieval Framework:** A relationship based ontology concept retrieval framework is proposed in Chapter 6 to address the research question **RQ5** i.e., What are the inherent characteristics of ontologies that can be used to rank relevant resources and ontologies for keyword queries? The framework uses a number of techniques to find and rank relevant concepts in an ontology for a given search term. Our proposed ranking model, DWRank is used as the core ranking model to rank matched results. Moreover, we proposed and implemented filters based on two strategies, the diverse results semantics and the intended type semantics. We investigated the extent to which these filters improve the performance of concept retrieval frameworks in general and how they might be adapted by other Semantic Web data retrieval techniques to improve their overall performance. The effectiveness of the framework is evaluated against a ground truth derived through a human evaluation. The evaluation results show that the concept retrieval framework outperforms the state-of-the-art ranking models as well as DWRank proposed in Chapter 6.

5. **Learning Concept Ranking for Ontology Search:** This contribution also addresses the third research question **RQ3** i.e., "How to rank relevant resources and ontologies for keyword queries?" We adopted a learning to rank approach to enhance the effectiveness of the concept ranking models in Chapter 7. The

ranking metrics, i.e. *Text relevancy*, *Hub Score* and *Auth Score*, defined in Chapter 6 are extended and used to learn a ranking model that combines these measures in a more effective way. The effectiveness of the proposed approach is measured by comparing the ranking produced by the proposed approach with DWRank [Butt et al., 2014b]. Moreover, a comprehensive comparison of the proposed approach with state-of-the-art ranking models is also presented. The results show that our proposed approach performs better than the best performing ranking algorithm for most queries. The maximum average `AP@10` on ten queries is 0.80 which is greater than the average of Tf-Idf, the best baseline ranking model, (i.e., 0.55). Moreover, the `AP@10` of the proposed approach ranges from 0.65 ~1.0, that means the performance is more robust on the ontology collection for the sample queries than the baseline ranking models.

6. **Ontology Recommendation for Structureless queries:** To address the research question **RQ4** i.e., "How to find the most relevant resources and ontologies that cover one or more resources users are interested in?" we proposed and implemented a framework that helps users in finding the best matching ontologies to a multi-keyword query in Chapter 8. Our approach recommends a ranked list of relevant ontologies using metrics that include the matching cost of a user query to an ontology, an ontology's informativeness and its popularity. These metrics are combined in a linear model to find the relevance score of an ontology to a query. The primary purpose of this approach is to improve the ontology recommendation for general queries on a domain independent ontology collection. For a comprehensive evaluation, we designed a multi-length keyword query that can be used for the evaluation of future ontology recommendation approaches. Finally, we investigate the performance of the proposed ranking model for multi-length queries by comparing it with the baseline approaches in ontology ranking through a user study. The evaluation of our approach against *AKTiveRank* and a linear version of our algorithm $RecOn_{ln}$ in a user study performed on the CBRBench dataset showed that $RecOn_{opt}$ outperforms the baseline state-of-the-art algorithm *AktiveRank* in 94% of the cases and $RecOn_{ln}$ in 92% of the cases. Moreover, $RecOn_{ln}$ outperforms *AKTiveRank* on all ten benchmark queries.

7. **Improving Ontology Ranking through the use of Optimization Technique:** This contribution is an extension of the previous contribution and addresses **RQ4**. We formalised ontology recommendation as an optimisation problem to recommend ontologies to the user that are as informative and popular as possible while incurring the least matching costs. We presented a methodology, in Section 8.3.2.4, for learning concept ranking where we discussed many of the decisions that need to be made when adapting learning to rank technique to rank concepts defined in ontologies. The purpose of this approach is to combine different evaluation metrics in a dynamic way to improve the ranking quality. We evaluated our approach in comparison to the linear baseline models in a user study. The results of our study prove the proposed approach is more

effective and scalable as compared to the linear baseline model.

9.2 Future Directions

The work presented in this thesis can be improved or extended in different ways as described in the following:

1. **Incorporate clustering to enable faceted browsing:** The keywords specified by a search engine's users may occur in different contexts and levels of detail. Consider for example the term '*travel*'. A user might not be aware of what exactly she wants to know about '*travel*'? A search engine typically returns long ordered list of results, but the user, in her limited amount of time, processes only the first few results. Thus a lot of truly important information hidden in the long result list will never be discovered. Ontology search becomes an exploration task and faces two challenges when the above situation occurs: how to effectively find and order matched results, and how to help the users explore an extensive set of ontologies that have been found. This thesis mainly focuses on the first challenge. The second challenge can be met by realising exploratory search on ontologies. Exploratory search is designed to serve complex and uncertain information needs, which is often the case in ontology search. It aims to help the user explore, process and interpret a large number of search results via continuous and exploratory interactions, mainly based on dynamically generated topics and clusters. To enable the faceted browsing for ontology search, the research needs to be conducted on identifying clusters and assigning meaningful topics to them.

2. **Extend ontology ranking models for ontology matching/merging:** The ontology retrieval techniques presented in this thesis consider a keyword query as an input and output a ranked list of the relevant ontologies and concepts for the query. One possible extension can be the techniques that retrieve the matched ontologies or ontology patterns from an ontology collection for an input ontology or ontology pattern. This can be extremely helpful for ontology matching or merging techniques that aim at aligning multiple ontologies. Such techniques merge two ontologies at a time and then select a matched ontology for a given ontology collection to further align newly selected ontology with the one of the previously aligned ontologies. An ontology retrieval framework that outputs ontologies for input ontologies can help choosing an ontology from a set of ontologies to be considered next for alignment. We believe that RecOn presented in Chapter 8 can be extended and evaluated for this purpose, and then the approach can be used by ontology matching techniques.

3. **Parallelise the indexing and ontology evaluation metrics computation:** The results presented in Chapter 6 and 8 confirm that using the proposed evaluation metrics for concept and ontology retrieval improves the quality of the ontology ranking but brings with it an associated cost in indices construction and query

response time. A possible solution to avoid the increase in index construction and query response time is the investigation of parallelized techniques for index construction and query processing steps in real-time. The index construction phase, where the *hub scores* and *auth scores* for each ontology in the ontology collection are computed and stored in indexes *ConHubIdx* and *OntAuthIdx* respectively, can be distributed over multiple processors since each record in the *ConHubIdx* is distinct and independent from other records in the index. Similarly *OntAuthIdx* can be build independent from *ConHubIdx*. In addition, querying the multiple records in the indexes can be built independently using multiple processors where a processor can be responsible for generating a set of evaluation metrics for a single candidate result. Since the query match step is computationally expensive, it can also be parallelized using multi-threading or multi-processing to generate the final list of matching records.

4. **Learn to rank from query log:** A learning to rank approach needs a gold standard with a set of good-quality query-answer pairs to have its features' weights tuned. The best practice for learning a ranking model is to employ a query log generated by real users. Since we did not have a real-world query log at hand, the ranking model is learnt using the gold standard available as a part of CBRBench. The performance of the proposed ranking model can be improved by learning the ranking model using a real query log in the form of a larger training dataset. We can investigate techniques for improving the learning to ranking model in the absence of a real query log. One of the potential solutions is incremental learning, where the initial model is learnt using a human created gold standard and then incrementally improved the performance of the ranking model with the future real-world query log as the users start querying through the proposed search engine.

5. **Improve Ontology Ranking Evaluation Framework:** In this thesis we established CBRBench [Butt et al., 2014a], an ontology ranking benchmark. It contains an ontology collection of a representative set of ontologies used on the Web, benchmark queries, and a gold standard established by human experts on the task of ranking ontology concepts for the benchmark queries. Moreover, performance evaluations of eight state-of-art ontology ranking models through these benchmarks are also presented in CBRBench. The CBRBench is an initial effort towards an ontology ranking benchmark; however, it can be improved in some aspects to make it more suitable ontology ranking evaluation framework. One of the most valuable improvements is to generate more meaningful multi-keyword queries and a gold standard for such queries. In Chapter 8, we extended CBRBench queries to get multi-keyword queries; however, a gold standard is still missing for such queries. Furthermore, the CBRBench constituents, i.e., ontology collection, benchmark queries, and the gold standard is publicly available along with the code for eight ranking model. We plan to regularly update the benchmark to add new ontologies, by retrieving newly register ontologies from prefix.cc, in the ontology collection. Similarly, recent

ranking models can be included and the performance evaluation of newly designed ranking models can be updated in the benchmark.

9.3 Concluding Remarks

In this thesis we have presented comprehensive research in ontology search and ranking techniques. First, we conducted an extensive survey of existing Semantic Web data retrieval techniques based on our taxonomy of such techniques. This taxonomy covers sixteen dimensions of Semantic Web retrieval techniques and categorised existing techniques along these dimensions. Based on our analysis, we have identified several shortcomings in existing techniques that led us to propose future research directions in this domain. We contributed by addressing some of the identified gaps in this thesis.

The primary contributions of this work are effective ontology ranking algorithms for domain independent ontology collections and single or multi keyword queries. We have proposed novel solutions for ontology and concept ranking, and evaluated the effectiveness of ontology ranking techniques. Specifically, we have proposed two concept ranking techniques (DWRank, and Concept ranking adapted from learning to rank approach) based on the centrality and authority of the concept in an ontology and an ontology collection respectively; and a relationship-based concept ranking framework that provides better matching concepts for user queries than the existing approaches. Moreover, an ontology recommendation approach to select and rank relevant ontologies against a multi-term structureless query are proposed. Our approach first finds a set of matched ontologies for a query string and then identifies the up to k most relevant matches using three measures, the *Matching cost*, the *Informativeness* and the *Popularity* of the matched ontologies. One of the approaches $RecOn_{ln}$ integrates these measures as a linear model and a second approach $RecOn_{opt}$ integrates these measures by formulating and solving it as an optimisation problem. Our evaluation shows that both approaches are more effective as compared to the baseline for an ontology recommendation; while $RecOn_{opt}$ outperforms $RecOn_{ln}$ in terms of its effectiveness and efficiency.

Another major contribution of our work relates to the evaluation of ontology ranking algorithms. Since a general framework for the evaluation of ontology ranking models has been missing in the literature, we have proposed a benchmark for ontology ranking with a standard set of measures. This benchmark suite for ontology ranking includes a collection of ontologies that was retrieved by crawling a seed set of ontology URIs derived from prefix.cc and a set of queries derived based on their popularity from a real query log from the Linked Open Vocabularies search engine. Further, it includes the results of the ideal ranking of the concepts in the ontology collection for the identified set of query terms established based on the opinions of ten ontology engineering experts. The ideal ranking is compared with eight state-of-the-art ranking algorithms and the precision at k, the mean average precision and the discounted cumulative gain is calculated for these algorithms rep-

resenting a baseline to compare an ontology ranking model. We put forward four recommendations, based on our analysis of the performance of ranking models, that we believe can considerably improve the performance of the discussed models for ranking resources in ontologies.

Bibliography

1998. Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering. http://www.cs.cmu.edu/~mccallum/bow/. [Online; accessed 12-November-2016]. (cited on page 48)

2010. TONES Ontology Repository. https://zenodo.org/record/32717. [Online; accessed 12-November-2016]. (cited on page 46)

Aasman, J., 2006. Allegro graph: Rdf triple database. *Cidade: Oakland Franz Incorporated*, (2006). (cited on page 25)

Abadi, D. J.; Marcus, A.; Madden, S. R.; and Hollenbach, K., 2007. Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd international conference on Very large data bases*, 411–422. VLDB Endowment. (cited on page 59)

Alani, H.; Brewster, C.; and Shadbolt, N., 2006. Ranking Ontologies with AKTiveRank. In *Proceedings of the International Semantic Web Conference (ISWC)*, 5–9. (cited on pages 3, 10, 11, 49, 61, 75, 79, 80, 81, 91, 92, 95, 114, 117, 119, 132, and 149)

Antoniou, G. and Van Harmelen, F., 2004. Web ontology language: Owl. In *Handbook on ontologies*, 67–92. Springer. (cited on page 20)

Anyanwu, K.; Maduko, A.; and Sheth, A., 2005. Semrank: Ranking complex relationship search results on the semantic web. In *Proceedings of the 14th international conference on World Wide Web*, 117–127. ACM. (cited on pages 51 and 62)

Anyanwu, K. and Sheth, A., 2003. P-queries: enabling querying for semantic associations on the semantic web. In *Proceedings of the 12th international conference on World Wide Web*, 690–699. ACM. (cited on page 51)

Ashburner, M.; Ball, C. A.; Blake, J. A.; Botstein, D.; Butler, H.; Cherry, J. M.; Davis, A. P.; Dolinski, K.; Dwight, S. S.; Eppig, J. T.; et al., 2000. Gene ontology: tool for the unification of biology. *Nature genetics*, 25, 1 (2000), 25–29. (cited on page 2)

Atkinson, M. P.; Bancilhon, F.; DeWitt, D. J.; Dittrich, K. R.; Maier, D.; and Zdonik, S. B., 1989. The object-oriented database system manifesto. In *DOOD*, vol. 89, 40–57. (cited on page 25)

Bailey, J.; Bry, F.; Furche, T.; and Schaffert, S., 2005. Web and semantic web query languages: A survey. In *Proceedings of the First international conference on Reasoning Web*, 35–133. Springer-Verlag. (cited on page 59)

Batzios, A.; Dimou, C.; Symeonidis, A. L.; and Mitkas, P. A., 2008. BioCrawler: An intelligent crawler for the semantic web. *Expert Systems with Applications*, 35, 1 (2008), 524–530. (cited on page 39)

Baumgartner, R.; Gatterbauer, W.; and Gottlob, G., 2009. Web data extraction system. In *Encyclopedia of Database Systems*, 3465–3471. Springer. (cited on page 59)

Bizer, C.; Heath, T.; and Berners-Lee, T., 2009a. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, (2009), 205–227. (cited on pages 2 and 22)

Bizer, C.; Lehmann, J.; Kobilarov, G.; Auer, S.; Becker, C.; Cyganiak, R.; and Hellmann, S., 2009b. Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web*, 7, 3 (2009), 154–165. (cited on page 23)

Bizer, C. and Schultz, A., 2009. The berlin sparql benchmark. *International journal on Semantic Web and information systems*, 5, 2 (2009). (cited on pages 43 and 59)

Bock, J.; Haase, P.; Ji, Q.; and Volz, R., 2008. Benchmarking owl reasoners. In *ARea-Workshop on Advancing Reasoning on the Web: Scalability and Commonsense*. Tenerife. (cited on page 40)

Bontas, E. P.; Mochol, M.; and Tolksdorf, R., 2005. Case studies on ontology reuse. In *Proc. of the 5th International Conference on Knowledge Management*. (cited on pages 2 and 3)

Brickley, D. and Miller, L., 2012. FOAF vocabulary specification 0.98. *Namespace document*, 9 (2012). (cited on page 2)

Broder, A., 2002. A taxonomy of web search. In *ACM Special Interest Group on Information Retrieval forum*, vol. 36, 3–10. ACM. (cited on page 59)

Broekstra, J.; Kampman, A.; and Van Harmelen, F., 2002. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *International semantic web conference*, 54–68. Springer. (cited on page 25)

Buitelaar, P.; Eigner, T.; and Declerck, T., 2004. OntoSelect: A dynamic ontology library with support for ontology selection. In *Proceedings of the Demo Session at the International Semantic Web Conference*. Citeseer. (cited on page 44)

Butt, A. S.; Haller, A.; and Xie, L., 2014a. Ontology search: An empirical evaluation. In *Proceedings of the International Semantic Web Conference*, 130–147. Riva del Gara, Italy. (cited on pages 3, 6, 11, 12, 44, 66, 71, 95, 96, 103, 108, 119, 121, 126, 132, 145, 156, and 161)

Butt, A. S.; Haller, A.; and Xie, L., 2014b. Relationship-based top-k concept retrieval for ontology search. In *The 19th International Conference on Knowledge Engineering and Knowledge Management*, 485–502. ACM. (cited on pages 11, 61, 62, 65, 146, and 159)

Butt, A. S.; Haller, A.; and Xie, L., 2015. A taxonomy of semantic web data retrieval techniques. In *Proceedings of the 8th International Conference on Knowledge Capture*, 9. ACM. (cited on page 4)

Butt, A. S.; Haller, A.; and Xie, L., 2016. DWRank: Learning concept ranking for ontology search. In *Semantic Web Journal*, vol. 7, 447–461. IOS Press. (cited on page 66)

Butt, A. S. and Khan, S., 2014. Scalability and performance evaluation of semantic web databases. *Arabian Journal for Science and Engineering*, 39, 3 (2014), 1805–1823. (cited on pages 44, 59, and 63)

Cantador, I.; Fernández, M.; and Castells, P., 2007. Improving ontology recommendation and reuse in webcore by collaborative assessments. (2007). (cited on page 49)

Carroll, J. J.; Dickinson, I.; Dollin, C.; Reynolds, D.; Seaborne, A.; and Wilkinson, K., 2004. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, 74–83. ACM. (cited on page 20)

Chang, C.-H.; Kayed, M.; Girgis, M. R.; and Shaala, K. F., 2006. A survey of web information extraction systems. *Knowledge and Data Engineering, IEEE Transactions on*, 18, 10 (2006), 1411–1428. (cited on page 59)

Cheng, G. and Qu, Y., 2013. Relatedness between vocabularies on the web of data: A taxonomy and an empirical study. *Web Semantics: Science, Services and Agents on the World Wide Web*, 20 (2013), 1–17. (cited on page 115)

Cheng, G.; Tran, T.; and Qu, Y., 2011. RELIN: Relatedness and informativeness-based centrality for entity summarization. In *International Semantic Web Conference*, 114–129. Springer. (cited on page 65)

Cheng, G.; Zhang, Y.; and Qu, Y., 2014. Explass: Exploring associations between entities via top-k ontological patterns and facets. In *International Semantic Web Conference*, 422–437. Springer. (cited on page 62)

Codd, E. F., 1970. A relational model of data for large shared data banks. *Communications of the ACM*, 13, 6 (1970), 377–387. (cited on page 25)

Cote, R. G.; Jones, P.; Apweiler, R.; and Hermjakob, H., 2006. The ontology lookup service, a lightweight cross-platform tool for controlled vocabulary queries. *BMC bioinformatics*, 7, 1 (2006), 97. (cited on page 44)

Dali, L.; Fortuna, B.; Duc, T. T.; and Mladenić, D., 2012. Query-independent learning to rank for rdf entity search. In *The Semantic Web: Research and Applications*, 484–498. Springer. (cited on pages 54 and 121)

d'Aquin, M. and Lewen, H., 2009. Cupboard — A Place to Expose Your Ontologies to Applications and the Community. In *Proceedings of the 6th European Semantic Web Conference* (Heraklion, Crete, Greece, 2009), 913–918. Berlin, Heidelberg. (cited on page 47)

d'Aquin, M. and Motta, E., 2011. Watson, More Than a Semantic Web Search Engine. *Semantic Web*, 2, 1 (2011), 55–63. (cited on pages 48 and 61)

Date, C. J., 2006. *An introduction to database systems*. Pearson Education India. (cited on pages 25 and 26)

David, C.; Olivier, C.; and Guillaume, B., 2012. A survey of RDF storage approaches. *ARIMA Journal*, 15 (2012), 11–35. (cited on page 59)

Delbru, R.; Toupikov, N.; Catasta, M.; Tummarello, G.; and Decker, S., 2010. Hierarchical link analysis for ranking web data. In *The Semantic Web: Research and Applications*, 225–239. Springer. (cited on page 54)

Ding, L.; Finin, T.; Joshi, A.; Pan, R.; Cost, R. S.; Peng, Y.; Reddivari, P.; Doshi, V.; and Sachs, J., 2004. Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, 652–659. ACM, New York, NY, USA. (cited on pages 3, 48, 49, 61, 63, and 132)

Ding, L.; Pan, R.; Finin, T.; Joshi, A.; Peng, Y.; and Kolari, P., 2005. Finding and ranking knowledge on the semantic web. In *Proceedings of the International Semantic Web Conference*, vol. 3729 of *Lecture Notes in Computer Science*, 156–170. (cited on pages 10, 95, and 149)

Erling, O. and Mikhailov, I., 2009. RDF support in the virtuoso dbms. In *Networked Knowledge-Networked Media*, 7–24. Springer. (cited on page 25)

Ferrara, E.; De Meo, P.; Fiumara, G.; and Baumgartner, R., 2014. Web data extraction, applications and techniques: A survey. *Knowledge-Based Systems*, 70 (2014), 301–323. (cited on page 59)

Fogaras, D., 2003. Where to start browsing the web? In *Innovative Internet Community Systems*, 65–79. (cited on page 100)

Franz, T.; Schultz, A.; Sizov, S.; and Staab, S., 2009. Triplerank: Ranking semantic web data by tensor decomposition. In *International semantic web conference*, 213–228. Springer. (cited on page 53)

Freeman, L. C., 1977. A set of measures of centrality based on betweenness. *Sociometry*, (1977), 35–41. (cited on pages 49 and 66)

Gangemi, A.; Catenacci, C.; Ciaramita, M.; and Lehmann, J., 2005. A theoretical framework for ontology evaluation and validation. In *Proceedings of the 2nd Italian Semantic Web Workshop*, vol. 166. (cited on page 75)

Giunchiglia, F.; Shvaiko, P.; and Yatskevich, M., 2004. S-Match: an algorithm and an implementation of semantic matching. In *European semantic web symposium*, 61–75. Springer. (cited on page 52)

Golbreich, C.; Horridge, M.; Horrocks, I.; Motik, B.; and Shearer, R., 2007. Obo and owl: Leveraging semantic web technologies for the life sciences. *The Semantic Web*, (2007), 169–182. (cited on page 44)

Goldberg, A. V. and Harrelson, C., 2005. Computing the shortest path: A search meets graph theory. In *Proceedings of the sixteenth annual ACM Society for Industrial and Applied Mathematics symposium on Discrete algorithms*, 156–165. (cited on page 66)

Gruber, T. R., 1995. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43, 5 (1995), 907–928. (cited on page 19)

Guarino, N. and Welty, C., 2002. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45 (2002), 61–65. (cited on page 75)

Guha, R., 2011. Introducing schema. org: Search engines come together for a richer web. *Google Official Blog*, (2011). (cited on page 6)

Guo, Y.; Pan, Z.; and Heflin, J., 2005. LUBM: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3, 2 (2005), 158–182. (cited on page 45)

Haarslev, V. and Möller, R., 2003. Racer: An owl reasoning agent for the semantic web. In *Proc. of the International Workshop on Applications, Products and Services of Web-based Support Systems, in conjunction with*, 91–95. (cited on page 39)

Haase, P.; Broekstra, J.; Eberhart, A.; and Volz, R., 2004a. A comparison of RDF query languages. In *International Semantic Web Conference*, 502–517. Springer. (cited on page 18)

Haase, P.; Broekstra, J.; Eberhart, A.; and Volz, R., 2004b. A comparison of rdf query languages. In *International Semantic Web Conference*, vol. 3298, 502–517. Springer. (cited on page 26)

Harth, A.; Kinsella, S.; and Decker, S., 2009. Using naming authority to rank data and ontologies for web search. In *International Semantic Web Conference*, 277–292. Springer. (cited on pages 53 and 54)

Harth, A.; Umbrich, J.; Hogan, A.; and Decker, S., 2007. YARS2: A Federated Repository for Querying Graph Structured Data from the Web. In *The Semantic Web* (Busan, Korea, 2007), 211–224. Springer-Verlag. (cited on page 63)

He, H.; Wang, H.; Yang, J.; and Yu, P. S., 2007. BLINKS: Ranked keyword searches on graphs. In *Proceedings of ACM SIGMOD international conference on Management of data*, 305–316. ACM. (cited on pages 55, 61, 63, 64, and 70)

Hepp, M., 2008. Goodrelations: An ontology for describing products and services offers on the web. In *Knowledge Engineering: Practice and Patterns*, 329–346. Springer. (cited on page 2)

Hodgson, R., 2009. oeGOV: Ontologies for e-Government. http://www.oegov.org. [Online; accessed 12-November-2016]. (cited on page 45)

Hogan, A.; Harth, A.; and Decker, S., 2006. ReConRank: A scalable ranking method for semantic web data with context. (2006). (cited on pages 51, 52, 65, and 70)

Hogan, A.; Harth, A.; Umbrich, J.; Kinsella, S.; Polleres, A.; and Decker, S., 2011. Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9, 4 (2011), 365–401. (cited on pages 11, 51, 52, 61, 62, 63, and 71)

Isele, R.; Umbrich, J.; Bizer, C.; and Harth, A., 2010. LDspider: An open-source crawling framework for the web of linked data. In *Proceedings of International Conference on Posters & Demonstrations*, 29–32. Citeseer. (cited on page 39)

Janik, M. and Kochut, K., 2005. Brahms: a workbench rdf store and high performance memory system for semantic association discovery. In *International Semantic Web Conference*, 431–445. Springer. (cited on page 25)

Joachims, T., 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 133–142. ACM. (cited on page 123)

Jonquet, C.; Musen, M. A.; and Shah, N. H., 2010. Building a biomedical ontology recommender web service. *Journal of biomedical semantics*, 1, 1 (2010), 1. (cited on pages 46 and 132)

Katsumi, M. and Grüninger, M., 2016. What is ontology reuse? In *FOIS*, 9–22. (cited on pages 1 and 2)

Khan, A.; Wu, Y.; Aggarwal, C. C.; and Yan, X., 2013. NeMa: Fast graph search with label similarity. *Proceedings of the Very Large Database Endowment*, 6, 3 (2013), 181–192. (cited on page 57)

Kleinberg, J. M., 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46, 5 (1999), 604–632. (cited on page 65)

Kothari, C. R., 2004. *Research methodology: Methods and techniques*. New Age International. (cited on page 8)

Laender, A. H.; Ribeiro-Neto, B. A.; da Silva, A. S.; and Teixeira, J. S., 2002. A brief survey of web data extraction tools. *ACM Sigmod Record*, 31, 2 (2002), 84–93. (cited on page 59)

Li, H., 2014. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 7, 3 (2014), 1–121. (cited on page 120)

Lozano-Tello, A. and Gomez-Perez, A., 2004. ONTOMETRIC: A method to choose the appropriate ontology. *Journal of Database Management*, 15, 2 (2004), 1–18. (cited on page 75)

Luo, Y.; Picalausa, F.; Fletcher, G. H.; Hidders, J.; and Vansummeren, S., 2012. Storing and indexing massive rdf datasets. In *Semantic Search over the Web*, 31–60. Springer. (cited on page 63)

Maedche, A.; Staab, S.; Stojanovic, N.; Studer, R.; and Sure, Y., 2001. Semantic portal-the seal approach. *Spinning the Semantic Web*, (2001), 317–359. (cited on page 70)

Martínez-Romero, M.; Vázquez-Naya, J. M.; Pereira, J.; and Pazos, A., 2014. BiOSS: A system for biomedical ontology selection. *Computer methods and programs in biomedicine*, 114, 1 (2014), 125–140. (cited on pages 6 and 47)

McBride, B., 2001. Jena: Implementing the rdf model and syntax specification. In *Proceedings of the Second International Conference on Semantic Web-Volume 40*, 23–28. CEUR-WS. org. (cited on page 25)

Meusel, R.; Mika, P.; and Blanco, R., 2014. Focused crawling for structured data. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 1039–1048. ACM. (cited on page 39)

Meymandpour, R. and Davis, J. G., 2013. Linked data informativeness. In *Web Technologies and Applications*, 629–637. Springer. (cited on page 65)

Miller, G. A., 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38, 11 (nov 1995), 39–41. (cited on page 92)

Ning, X.; Jin, H.; and Wu, H., 2008. RSS: A framework enabling ranked search on the semantic web. *Information Processing & Management*, 44, 2 (2008), 893–909. (cited on page 52)

Noy, N. F.; Alexander, P. R.; Harpaz, R.; Whetzel, P. L.; Fergerson, R. W.; and Musen, M. A., 2013. Getting lucky in ontology search: a data-driven evaluation framework for ontology ranking. In *International Semantic Web Conference*, 444–459. (cited on page 46)

Noy, N. F. and d'Aquin, M., 2012. Where to Publish and Find Ontologies? A Survey of Ontology Libraries. *Web Semantics: Science, Services and Agents on the World Wide Web*, 11 (2012), 96–111. (cited on pages 1, 3, 4, 10, 44, 60, 70, 95, and 132)

Noy, N. F.; McGuinness, D. L.; et al., 2001. Ontology development 101: A guide to creating your first ontology. (cited on page 1)

Noy, N. F.; Shah, N. H.; Whetzel, P. L.; Dai, B.; Dorf, M.; Griffith, N.; Jonquet, C.; Rubin, D. L.; Storey, M.-A.; Chute, C. G.; and Musen, M. A., 2009. BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic acids research*, (2009), gkp440. (cited on pages 3, 10, 46, 61, 66, 69, 70, 95, and 132)

Oren, E.; Delbru, R.; Catasta, M.; Cyganiak, R.; Stenzhorn, H.; and Tummarello, G., 2008. Sindice. com: A document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3, 1 (2008), 37–52. (cited on pages 50, 51, 53, 54, 61, 62, 63, 70, and 71)

Oren, E.; Delbru, R.; and Decker, S., 2006. Extending faceted navigation for rdf data. In *The Semantic Web-ISWC*, 559–572. Springer. (cited on page 61)

Owens, A., 2009. An investigation into improving rdf store performance. (2009). (cited on page 23)

Page, L.; Brin, S.; Motwani, R.; and Winograd, T., 1998. The PageRank citation ranking: Bringing order to the Web. In *Proceedings of the 7th International World Wide Web Conference*, 161–172. Brisbane, Australia. (cited on pages 11, 48, 65, 80, and 99)

Pan, J. Z.; Thomas, E.; and Sleeman, D., 2006. Ontosearch2: Searching and querying web ontologies. *Procedings of International Conference on World Wide Web*, (2006), 211–218. (cited on page 45)

Patel, C.; Supekar, K.; Lee, Y.; and Park, E., 2003. OntoKhoj: a semantic web portal for ontology searching, ranking and classification. In *Proceedings of the fifth ACM international workshop on Web information and data management*, 58–61. ACM. (cited on page 48)

Pound, J.; Mika, P.; and Zaragoza, H., 2010. Ad-hoc Object Retrieval in the Web of Data. In *Proceedings of the 19th International Conference on World Wide Web* (Raleigh, North Carolina, USA, 2010), 771–780. ACM, New York, NY, USA. (cited on pages 83 and 84)

Rada, R.; Mili, H.; Bicknell, E.; and Blettner, M., 1989. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19, 1 (1989), 17–30. (cited on page 49)

Robertson, S. E.; Walker, S.; Jones, S.; Hancock-Beaulieu, M. M.; Gatford, M.; et al., 1995. Okapi at TREC-3. *National Institute of Standards & Technology Special Publication*, 109 (1995), 109–109. (cited on pages 65 and 78)

Sabou, M.; Lopez, V.; and Motta, E., 2006. Ontology selection for the real semantic web: How to cover the queen's birthday dinner? In *Managing Knowledge in a World of Networks*, 96–111. Springer. (cited on page 49)

Salton, G. and Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24, 5 (1988), 513–523. (cited on pages 65 and 77)

Salton, G.; Wong, A.; and Yang, C.-S., 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18, 11 (1975), 613–620. (cited on pages 65 and 78)

Sarawagi, S., 2008. Information extraction. *Foundations and trends in databases*, 1, 3 (2008), 261–377. (cited on page 59)

Schmidt, M.; Hornung, T.; Lausen, G.; and Pinkel, C., 2009. Spˆ 2bench: a sparql performance benchmark. In *International Conference on Data Engineering*, 222–233. IEEE. (cited on pages 43 and 59)

Shannon, C. E., 2001. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5, 1 (2001), 3–55. (cited on page 65)

Sirin, E.; Parsia, B.; Grau, B. C.; Kalyanpur, A.; and Katz, Y., 2007. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5, 2 (2007), 51–53. (cited on page 39)

Smith, B.; Ashburner, M.; Rosse, C.; Bard, J.; Bug, W.; Ceusters, W.; Goldberg, L. J.; Eilbeck, K.; Ireland, A.; Mungall, C. J.; et al., 2007. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25, 11 (2007), 1251–1255. (cited on pages 45 and 61)

Spanoudakis, G. and Constantopoulos, P., 1993. Similarity for analogical software reuse: A conceptual modelling approach. In *Advanced Information Systems Engineering*, 483–503. Springer. (cited on page 49)

Staab, S. and Studer, R., 2013. *Handbook on ontologies*. Springer Science & Business Media. (cited on pages 1, 5, and 19)

Stegmaier, F.; Gröbner, U.; Döller, M.; Kosch, H.; and Baese, G., 2009. Evaluation of current rdf database solutions. In *Proceedings of the 10th International Workshop on Semantic Multimedia Database Technologies (SeMuDaTe), 4th International Conference on Semantics And Digital Media Technologies (SAMT)*, 39–55. (cited on page 31)

Stocker, M.; Seaborne, A.; Bernstein, A.; Kiefer, C.; and Reynolds, D., 2008. Sparql basic graph pattern optimization using selectivity estimation. In *Proceedings of the 17th international conference on World Wide Web*, 595–604. ACM. (cited on page 31)

Tartir, S. and Arpinar, I. B., 2007. Ontology evaluation and ranking using ontoqa. In *International Conference on Semantic Computing*, 185–192. IEEE. (cited on page 49)

Thomas, E.; Pan, J. Z.; and Sleeman, D., 2007. ONTOSEARCH2: Searching ontologies semantically. In *Workshop on OWL: Experiences and Directions*, vol. 258 of *CEUR Workshop Proceedings*. (cited on page 45)

Trotman, A., 2005. Learning to rank. *Information Retrieval*, 8, 3 (2005), 359–381. (cited on page 66)

Tummarello, G.; Cyganiak, R.; Catasta, M.; Danielczyk, S.; Delbru, R.; and Decker, S., 2010. Sig. ma: Live views on the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8, 4 (2010), 355–364. (cited on pages 11, 51, 61, and 62)

Vandenbussche, P.-Y. and Vatant, B., 2014. Linked Open Vocabularies. *ERCIM news*, 96 (2014), 21–22. (cited on pages 3, 10, 47, 61, 69, 82, 95, 132, and 146)

Vrandečić, D. and Krötzsch, M., 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57, 10 (2014), 78–85. (cited on page 23)

Wu, G.; Li, J.; Feng, L.; and Wang, K., 2008a. Identifying potentially important concepts and relations in an ontology. In *International Semantic Web Conference*, 33–49. (cited on pages 100 and 114)

Wu, Q.; Burges, C. J.; Svore, K. M.; and Gao, J., 2008b. Ranking, boosting, and model adaptation. Technical report, Microsoft Research. (cited on page 124)

Wu, Y.; Yang, S.; and Yan, X., 2013. Ontology-based subgraph querying. In *International Conference on Data Engineering*, 697–708. IEEE. (cited on pages 57, 61, 64, and 70)

Yan, X.; Yu, P. S.; and Han, J., 2005. Substructure similarity search in graph databases. In *Proceedings of ACM SIGMOD international conference on Management of data*, 766–777. ACM. (cited on pages 55, 61, and 64)

Yang, S.; Wu, Y.; Sun, H.; and Yan, X., 2014. Schemaless and structureless graph querying. *Proceedings of the Very Large Database Endowment*, 7, 7 (2014), 565–576. (cited on pages 57, 61, 64, and 70)

Yuan, D. and Mitra, P., 2011. A lattice-based graph index for subgraph search. In *WebDB*. (cited on page 56)

Yuan, D. and Mitra, P., 2013. Lindex: a lattice-based index for graph databases. *The Very Large Database*, 22, 2 (2013), 229–252. (cited on pages 56, 61, and 64)