ANGLIA RUSKIN UNIVERSITY

AN INTEGRATED ARCHITECTURE FOR SEMANTIC SEARCH

AROOJ FATIMA

A thesis in partial fulfilment of the requirements of Anglia Ruskin University for the degree of Doctor of Philosophy

Submitted: June 2016

Dedicated to my late sister *Maria Shams*. May her soul rest in peace.

Acknowledgements

This dissertation was prepared in part fulfilment of the requirements of the degree of Doctor of Philosophy under the supervision of Dr. Cristina Luca and Dr. George Wilson at Anglia Ruskin University. This Ph.D. journey has been a truly life-changing experience for me and it would not have been possible without the support that I have received from many people.

I would like to express my special gratitude and thanks to my first supervisor Dr. Cristina Luca, she have been a fabulous mentor for me. I would like to thank her for all the encouragement and the trust she showed in me. I would like to show my sincere appreciation to my second supervisor Dr. George Wilson for his motivation, enthusiasm, and immense knowledge. I would also like to thank my head of department, Professor Marcian Cirstea, for his guidance and advice on both research as well as on my career.

I gratefully acknowledge the funding I received for my Ph.D. from Anglia Ruskin University. I take this opportunity to express gratitude to all of the Department faculty members for their help and support. Thanks to Dr. Mike Hobbs for his encouragement and supportive behaviour whenever I needed some advice. Thanks also to Tim Reynolds for his valuable comments and suggestions in my annual review meetings.

A very special thanks to my family. Words cannot express how grateful I am to everyone in my family for all their prayers and well wishes for me especially my younger brother Husnain Virk for being always there to listen to me. I would also like to thank all of my friends Imran Mahmood Khan Dhuddi, Vanitha Bhaskaran and Tanja Fitze who always believed in me and have been a big moral support to me. I can never forget to express my gratitude to my dearest friend Kathleen Dean who has always been a source of strength and wisdom to me.

A very special thanks to Dr. Nicholas Caldwell from University Campus Suffolk for his valuable input and feedback on my research and for giving his

precious time to discuss research ideas with me. I highly acknowledge the help that I received from my colleague Jonathan Kimmitt who guided me in many ways throughout these three years of my Ph.D. I am also very grateful to my other colleagues in BRY115 especially Mark Graham and Mohamed Salah Kettouch for the fruitful discussions we always had at our desks. Of course a big thank you to Robert Waixel, for helping me doing the final proof-reading of this thesis.

Last but by no means least, I would like to give an expression of appreciation to my husband Abdul Wahhab Khan who stood by me throughout this venture and was always there to encourage, listen and discuss my research.

ANGLIA RUSKIN UNIVERSITY

ABSTRACT

FACULTY OF SCIENCE AND TECHNOLOGY

DOCTOR OF PHILOSOPHY

AN INTEGRATED ARCHITECTURE FOR SEMANTIC SEARCH

AROOJ FATIMA

JUNE 2016

Linked Open Datasets are one solution to the problem of presenting data in a structured and meaningful manner such that software agents can search, reason with and manipulate this data based on an understanding of its semantics. Accessing structured data from Linked Open Datasets currently requires the use of formal query languages (such as SPARQL) which poses significant difficulties for the end users.

One way to solve this problem is to provide a Natural Language Interface (NLI) to query semantic data. The author undertook a comprehensive literature survey of existing semantic search tools and performed a critical analysis to identify their strengths and weaknesses. Although some of the existing tools support natural language, they are limited in their techniques for query processing, result ranking, result readability and ease of integration with other search tools. Based upon this analysis, this research proposes a new architecture framework called SIRF (**S**emantic **I**nformation **R**etrieval **F**ramework) for semantic search to address these shortcomings.

This thesis provides a complete overview of the proposed framework, including: the research challenges it addresses; its architecture; the techniques to map user queries to SPARQL queries and to rank domains based on ontology concepts; and the evaluation of the proposed system through a prototype. Evaluation of the prototype demonstrated the validity of the approach. However the quality of resulting queries (and consequently retrieved answers) depended upon the accuracy of the NLP parsers invoked by the prototype. Syntactically well structured NL queries were correctly parsed, yielding better formed SPARQL queries. Less structured NL queries performed poorly. As the framework is not tied to any particular parser, result quality can be improved by utilising better parsers as they become available.

The author believes that this work can be employed by a variety of end-user applications that wish to utilise structured data.

Keywords: linked open data, ontologies, semantic research, information retrieval, SPARQL, natural language interfaces

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

$\Leftrightarrow$	Implies that.
$\cap$	A set of common elements.
$:=$	Defined as.
$\emptyset$	An empty set.
$\forall$	For all.
$\leftarrow$	Assigned to.
$>$	Greater than.
$\in$	A sub set of OR belong to.
$<$	Less than.
$\neq$	Not equal to.
$\notin$	Not in.
$\rightarrow$	Object value.

List of Abbreviations

API Application Program Interface

ATC Answer Type Calculator

BL British Library

BNB British National Bibliography

CAT Calculated Answer Type

CPI Class, Property and Instance

DSD Data-set Dispatcher

LOD Linked Open Data

NE Named Entity

NER Named Entity Recognition

NL Natural Language

NLIs Natural Language Interfaces

NLP Natural Language Processing

OWL Web Ontology Language

RDF Resource Description Framework

SEO Search Engine Optimisation

URI Uniform Resource Identifier

URL Uniform Resource Locator

W3C World Wide Web Consortium

WWW World Wide Web

XML Extensible Markup Language

List of Acronyms

CKAN Comprehensive Knowledge Archive Network

GATE General Architecture for Text Engineering

JAWS Java API for WordNet Searching

SIRF Semantic Information Retrieval Framework

SPARQL SPARQL Protocol and RDF Query Language

List of Related Publications

- Fatima, A., Luca, C. and Wilson, G., 2014. New Framework for Semantic Search Engine. 16th International Conference on Computer Modelling and Simulation (UKSIM-2014). March 2014. Paper No. 1569916557.
- Fatima, A., Luca, C. and Wilson, G., 2014. User Experience and Efficiency for Semantic Search Engine. 14th International Conference on Optimization of Electrical and Electronic Equipment (OPTIM 2014). May 2014. Paper RD-005193.
- Hobbs, M., Luca, C., Fatima, A., Warnes, M., 2014. Ontological analysis for dynamic data model exploration. Electronic Journal of Applied Statistical Analysis: Decision Support Systems and Services Evaluation, North America, 5, DEC. 2014.
- Fatima, A. Luca, C. Wilson, G. Kettouch, M. S., 2015. Result Optimisation for Federated SPARQL queries (UKSIM 2015). 25-27 March 2015.
- Kettouch, M. S. Luca, C. Hobbs, M. Fatima, A., 2015. Data Integration Approach for Semi-structured and Structured Data (Linked Data). Industrial Informatics 2015 (INDIN) IEEE International Conference on 22-24 July 2015.
- Fatima, A. Luca, C. Hobbs, M., 2015. SPARQL query from free text user query. Industrial Informatics 2015 (INDIN) IEEE International Conference on 22-24 July 2015.
- Fatima, A. Luca, C. Wilson, G., 2015. User queries for Semantic Search. International Journal of Simulation, Systems, Science and Technology (IJSSST) V 16.

AN INTEGRATED ARCHITECTURE FOR SEMANTIC SEARCH

AROOJ FATIMA

COPYRIGHT

"If information is power and riches, then it is not the amount that gives the value, but access at the right time and in the most suitable form".[1]

[1] General Architecture for Text Engineering (https://gate.ac.uk/ie/)

Chapter 1

Introduction

1.1 Motivation

The exchange of data for interoperability is an essential feature for distributed and open knowledge systems. If two or more systems are able to communicate and exchange data, they exhibit syntactic interoperability. This kind of interoperability requires some common standards to communicate e.g. XML[2] or relational databases. Application Program Interfaces (APIs) are a good example of syntactic interoperability. On the other hand, these standards exchange data without knowing its meaning, hence results are not very useful and require much human effort to manipulate data. The ability of one system to automatically interpret the information from another system is called semantic interoperability. Broadly defined, semantic interoperability enables one IT system to receive information from another IT system and allows it to apply its business rules against the information received [Dixon *et al.*, 2014]. This semantic interoperability again needs a common standard/protocol to define meanings of data and resolve any potential ambiguities. The current technology to achieve this purpose is ontology.

There is a large amount of data stored over the Internet that is only useful if accessed as meaningful information [El-gayar *et al.*, 2015]. Yet even to access data from the Internet we need a smart search facility. Search engines are the tools of choice to help users find data from the World Wide Web. To extract data, most of the search engines use syntax-based search or full-text search methods. Full-text searching is a technique whereby a computer program matches terms in a search query with terms embedded within individual documents in a database [Beall, 2008]. An important issue with the full-text search is that it relies on linguistic matching that is, matching a word or phrase in a search

[2]Extensible Markup Language (XML) - is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

query with the same word or phrase in a document in the database. This kind of search is subject to failure when a variant term exists [Beall, 2010].

Modern syntax based search engines use various techniques to compensate for the limitations of full-text search e.g. auto-correct and auto-suggest [Jurczyk, 2014], result ranking (based on relevancy factors) and content score [Selvan and Dharshini, 2012]. Search Engine Optimisation (SEO) is another technique that can be defined as "the process of editing a web site's content and code in order to improve visibility within one or more search engines" [Killoran, 2013, p. 53]. Keywords, meta-tags and micro-formats are the main tools used for SEO. These techniques enhance the factor of user friendliness and increase the chances of more accurate results but these are not the ultimate solution. That is why the data searched by a syntax-based search engine has a number of limitations, including high recall with low precision (e.g. thousands of results in response to one or few keywords), low or no recall (when there is no relevant results), and a high sensitivity of results to vocabulary [Antoniou and Harmelen, 2008].

The World Wide Web (WWW) was originally designed as an information space, with the goal that it should be useful not only for human-human communication, but also machines should be able to read and understand data [Berners-Lee, 1998]. Unfortunately the idea could not be fully adopted and internet information was collected and processed in a way to be used by humans only. Machines will simply get data from a request and pass it to some query that will return all possible matches to that string pattern. The progress in web applications was so rapid and promising that the real idea that the World Wide Web was all about, was overlooked.

A semantic web is the optimised solution to overcome the problems of these challenges. The Semantic Web can be described as a web of documents linked in such a way so that the data becomes readable and understandable to machines in a meaningful way, hence allowing them to intelligently match related data [Berners-Lee, 1999]. One way of viewing this Semantic Web is that semantically searching the World Wide Web returns results relevant to the *meaning* of the search query. It is neither exclusively a new way of storing data nor exclusively a new way of querying data, but rather a hybrid approach relevant to existing data whilst supporting new ways of storing data and allowing the utilisation of the semantics of that data to improve the quality of search results.

In recent years, the Resource Description Framework (RDF) [Klyne and Carroll, 2006] has become a popular protocol for storing web-based data with well-defined meanings. Linked Open Data (LOD) [Bizer, 2009] is published structured data in RDF format that can be connected to other LOD datasets. The concept of Linked Open Data has even widened the scope of RDF. Whilst RDF data is routinely used by many organisations

(e.g. gov.uk and bbc.co.uk) its potential to improve semantic searches is now of interest to the Database and Internet research communities. RDF will be explained in detail in Chapter 2.

1.2 Problem Statement

The progress of the Semantic Web can be witnessed by the increased volume of structured and linked data. The volume of Linked Open Datasets has increased from a dozen datasets in 2007 to hundreds of datasets by 2014 [Schmachtenberg *et al.*, 2014]. The growth of LOD has opened tremendous opportunities to change the form of the way web search works. Although the Semantic Web is already used in practice, there is a need for a strong semantic search system to utilise the full potential of that web. A number of efforts have been made to investigate and implement semantic search tools (i.e. FREyA [Damljanovic *et al.*, 2011], PowerAqua [Lopez *et al.*, 2011] and QuestIO [Damljanovic *et al.*, 2008]) but yet there remains a gap between end user interface and semantic datasets. The Semantic Web should be equally accessible by computers using specialized languages and humans using natural language [Katz *et al.*, 2002]. To find the currency of a country (say Pakistan), an end user may write a query "what is the currency of Pakistan" whereas the semantic datasets must be accessed by a SPARQL[3] query that can be written as follows

```
SELECT DISTINCT ?currency
WHERE{
res:Pakistan dbo:currency ?currency
}
```

Natural Language Interfaces (NLIs) are a promising option for casual end users to interact with semantic knowledge bases [Kaufmann and Bernstein, 2007]. Several projects (such as [Frank *et al.*, 2007], [Popescu *et al.*, 2003] and [Thompson *et al.*, 2005]) have shown that NLIs can perform well in retrieval tasks without being unnecessarily complex. To make semantic search usable for the end users, a semantic search tool should be able to translate a user query to an equivalent SPARQL query.

An effective user interface[4] that accepts natural language queries to fetch information from LOD datasets is still a challenge for the Semantic Web search communities. To build

[3]SPARQL Protocol and RDF Query Language - is a W3C recommendation to query semantic data stored in RDF format (https://www.w3.org/TR/rdf-sparql-query/).

[4]A user interface, in this context, refers to an interface for a search application that can be accessed by the end users.

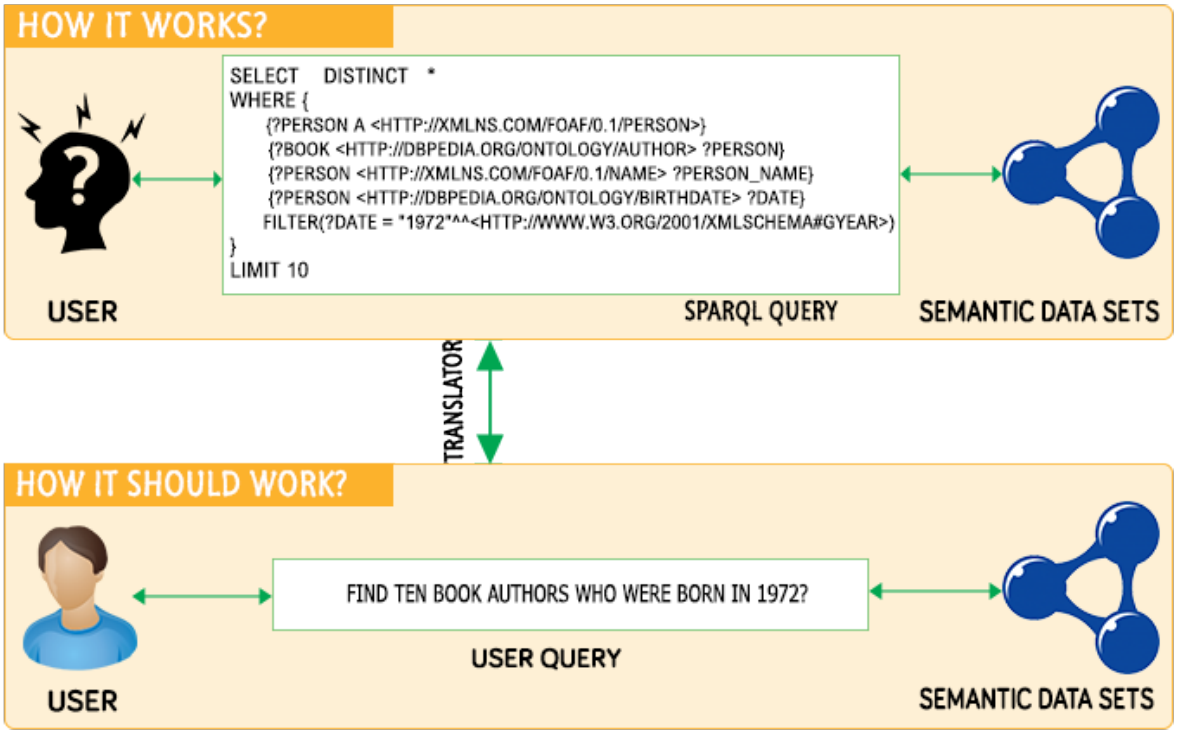


Figure 1.1: Problem Statement (Source: Author, 2015)

such a system requires interoperation among various semantic technologies, knowledge resources and a middleware service that translates the user query to a semantic query. Therefore, there is a need to create an appropriate framework to serve this purpose.

1.3 Research Aim and Scope

The work researched in this thesis has been driven by the above problem statement as shown in Figure 1.1. The overall aim of this research is

> To improve the effectiveness of semantic search technologies for the end users.

To narrow down the scope of the above broader aim, this thesis focuses on finding a way of semantic data retrieval using a human friendly language such as English. The main focus of the research is to facilitate the end users in exploiting semantic data whilst avoiding the complexity of semantic technologies.

This work will be of interest to the developers (who need semantic search interfaces for their RDF datasets), end users (who intend to query semantic datasets for different purposes i.e. research, statistics etc.) and the search tool providers (who want to embed a semantic search capability).

1.4 Research Questions

The core research question for this thesis is

> *How to design a semantic search system that makes semantic data accessible and eliminates the problems of existing systems?*

The above broader question has been divided into following more focused questions as follows:

1. What are the limitations of existing semantic search tools?
2. What are the features that can improve the effectiveness of a semantic search tool?
3. Considering the existing semantic data technologies (i.e. RDF, Ontologies and SPARQL), is it feasible to design an effective semantic search system that can be integrated with other search tools?

1.5 Research Hypothesis

The researcher believes that it is possible to design an effective semantic search system (using existing semantic technologies) that supports natural language queries and facilitates end users to access linked open datasets, hence improving the accessibility of semantic data.

1.6 Original Contributions to Knowledge

This thesis makes a practical as well as theoretical contribution to the area of semantic search, bridging the gap between the end users and the search system to improve the accessibility. The detail of the contributions are as follows

- Analysis of the state-of-art for semantic search tools. This analysis attempts to provide an understanding of the different search approaches and the challenges faced by them.
- The introduction of SIRF - the author's proposed framework for semantic search. The framework acts as a layer between the user interface and linked open datasets and delivers an effective semantic search experience. SIRF can be used as a backbone of the search tool for any system using semantic web technologies (e.g. Linked Open

Data (LOD) and a SPARQL endpoint to access LOD data). SIRF contains a number of modules that may work independently with minimal configuration including ontology processor, query optimiser, query formatter, API and other components. This flexibility enables the framework to be used as a whole or as a part of other search tools.

- User query translation. The findings and new algorithms for query translation aim to facilitate semantic search tools in supporting natural language queries.

1.7 Approach

The approach this thesis takes in solving the problem of user query translation to SPARQL query, is divided in four main parts (see Figure 1.2 for detailed design approach).

- Data collection (from literature, research communities, available tools)
- Analysis of the state-of-art for the existing semantic search system
- Proposal of a solution to the problems identified (in the step above)
- Evaluation of the solution with a prototype

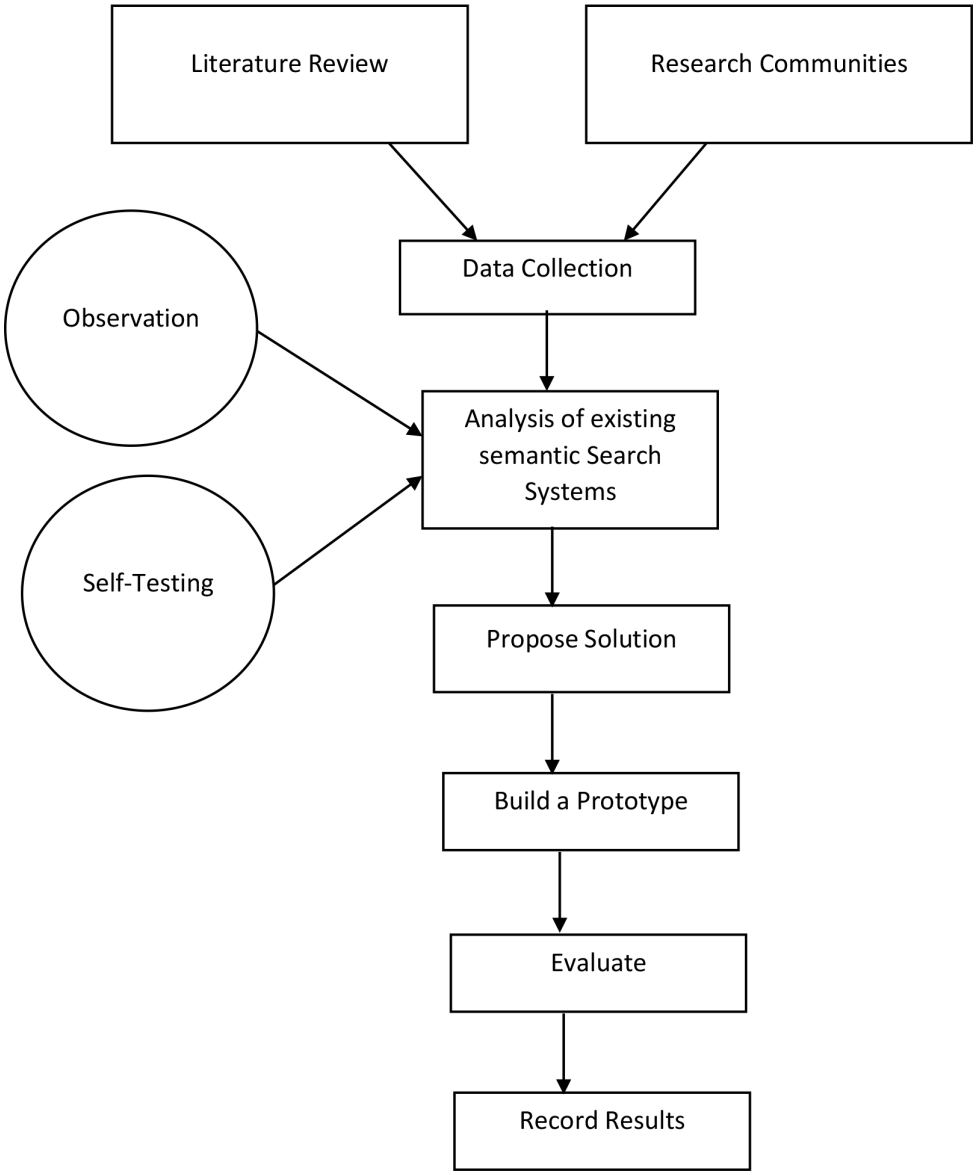


Figure 1.2: Research Approach (Source: Author, 2015)

1.8 Thesis Structure

The remainder of the thesis has been arranged as follows:

Chapter 2: Drawing extensively from the literature, this chapter explains the background of the Semantic Web and its technologies. It covers all the preliminaries to be used in upcoming chapters i.e. RDF, ontologies, SPARQL and OWL. The chapter also explores the challenges faced by semantic web search interfaces.

Chapter 3: This explains the conceptual framework and methodology used through-

out the thesis.

Chapter 4: Following the introductory chapters, this gives an analysis of the related work and highlights the gaps that need to be filled. This chapter also explores the criteria for a semantic search system that enables end users and other search systems to access structured data using natural language queries.

Chapter 5: After the analysis and evaluation of the related work, Chapter 5 introduces SIRF (the proposed solution), namely a new framework for the semantic search. The chapter describes the individual modules in the framework and explains their usage for information retrieval.

Chapter 6: This elaborates on the modules of SIRF that take part in building the Knowledge Base of the proposed system. This chapter further highlights the importance of caching for quick information retrieval. The chapter also introduces the algorithm for ontology processing.

Chapter 7: This explains the procedures for text parsing that are the initial steps towards SPARQL query generation. The chapter highlights the requirement of natural language processing and describes SIRF's modules involved in syntax processing i.e. the Syntax Analyser and the Semantic Analyser. The chapter introduces an algorithm for dependency processing to produce simplified syntactic pairs that improves the performance of the proposed system to find semantic relations.

Chapter 8: The core of Chapter 8 is the process of converting natural language user queries to SPARQL queries. This chapter explains the syntax of SPARQL and its mapping to system generated variables, as well as the procedures to handle queries with negation, conjunctions and quantification. The chapter also introduces the algorithms for query translation and finding the answer type of a query.

Chapter 9: Chapter 9 focuses on the Result Optimisation module of SIRF. The chapter describes the proposed procedures to rank results and improve result readability.

Chapter 10: Chapter 10 describes the implementation of a prototype for the proposed framework. This further provides evaluation of the proposed framework (using a prototype) based on the criteria defined in Chapter 4. The chapter records the results from this phase that help to identify the limitations.

Chapter 11: This gives a summary of research findings and highlights future directions for research based on the limitations outlined in Chapter 10.

Chapter 2

Background

2.1 Introduction

The first half of this chapter defines the semantic web and its core technologies emphasising the concepts that will be used in later chapters. The second half of the chapter describes the concept of semantic search and explains the challenges it confronts.

2.2 The Semantic Web

The term 'semantics' is used in many different contexts but the most appropriate corresponding English term is 'meaning' [Studer *et al.*, 1998]. Tim Berners-Lee [Berners-Lee, 1999, p. 169] had the following vision for the Semantic Web

"I have a dream for the Web in which computers become capable of analysing all the data on the Web; the content, links, and transactions between people and computers. A Semantic Web, which should make this possible, has yet to emerge."

The Semantic Web is rapidly growing. The number of Linked Open Datasets has been increased by 271% from 2011 to 2014 (see Table 2.1). Much research has been done to improve the core technologies of the Semantic Web architecture (as detailed in Section 2.3). The semantic research communities are also working to develop new technologies to support and improve the existing ones. Broadly speaking, there seem to be several research directions that are actively being studied i.e. data mining, information retrieval, and information extraction [Nica *et al.*, 2015].

Category	Percentage	Datasets 2011 (294)	Datasets 2014 (1091)	Growth (271%)
Social Networking	48%	-	520	-
Government	18%	49	199	306%
Publications	13%	87	138	59%
Life sciences	8%	41	85	107%
User-generated Content	5%	20	51	155%
Cross-domain	4%	41	47	15%
Media	2%	25	24	-4%
Geographic	2%	31	27	-13%

Table 2.1: A comparison of Linked Open Datasets' growth from 2011 to 2014 (Source: [Schmachtenberg *et al.*, 2014])

2.3 Semantic Web Technologies

"Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data" [W3C, 2011]. Figure 2.1 describes the Semantic Web Architecture.

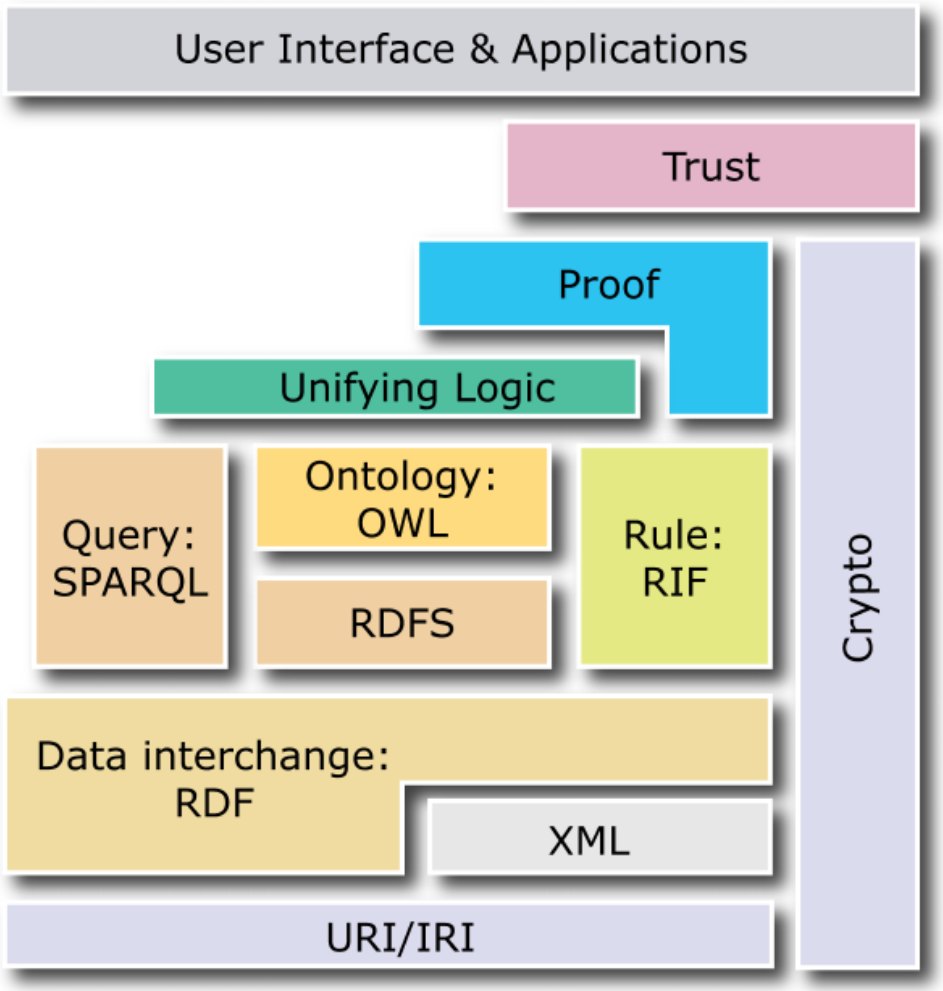


Figure 2.1: Semantic Web Architecture (Source: [Berners-Lee, 2006])

At the core of any information retrieval system is data, and how it can be stored such that it can be extracted for multiple purposes. In the same way the Semantic Web also requires two types of data handling techniques:

- To store data in an universal structure
- To extract data from store points

To store data in a universal standard format the World Wide Web Consortium (W3C[5]) introduced a combination of technologies. The technologies relevant to this work are as follows

- Resource Description Framework (RDF)
- Ontology
- Web Ontology Language (OWL)

To extract data from RDF datasets, W3C introduced a query language called

- SPARQL

2.3.1 Resource Description Framework (RDF)

RDF[6] is a standard recommended by W3C to store data with pre-defined meanings. RDF documents can be represented in different syntax formats including N-Quads[7], N-Triples[8], Turtle[9], JSON-LD[10], Notation3 [11] and RDF/XML[12]. The fundamental concepts of storing data in RDF documents are resources, properties, statements and graphs [Antoniou and Harmelen, 2008]. Resource Description Framework Schema (RDFS) is a vocabulary that extends RDF for describing properties and classes for resources [He *et al.*, 2011].

[5]http://www.w3.org/ - The World Wide Web Consortium is the main international standards organization for the World Wide Web.
[6]http://www.w3.org/rdf - Resource Description Framework.
[7]http://www.w3.org/TR/n-quads/ - N-Quads is a format to represent RDF.
[8]http://www.w3.org/TR/n-triples/ - N-Triples is a format to represent RDF.
[9]http://www.w3.org/TR/turtle/ - Turtle is a format to represent RDF.
[10]http://www.w3.org/TR/json-ld/ - JSON-LD is a format to represent RDF.
[11]http://www.w3.org/TeamSubmission/n3/ - Notation3 is a format to represent RDF.
[12]http://www.w3.org/TR/rdf-syntax-grammar/ - RDF/XML is the most used format to represent RDF data.

2.3.1.1 Resources/Classes

Anything defined in RDF is called an RDF resource (e.g. Lecturer, Student, and Country). Resources can also be identified as classes. All RDF resources are subclasses of rdfs:Resource class that is a class of everything. Classes are uniquely identified by Internationalised Resource Identifiers(IRIs [13]). The members of a class are known as *instances* of the class that are normally linked to the class using properties e.g. "George" is an instance of the class "Person" that is linked using the property "name".

2.3.1.2 Properties

Properties (in RDF) define a relation between subject resources and object resources [Antoniou *et al.*, 2012] (e.g. taught by, located in). Figure 2.2 explains how a subject resource (Arooj) is related to an object resource (Cambridge) using a property (livesIn).

2.3.1.3 Statements

An RDF statement represents an RDF triple. A triple in RDF contains three components

1. Subject - is an RDF resource that can be a Uniform Resource Identifier (URI) or a blank node[14].
2. Predicate - is a relation (i.e. property) between subject resource and object resource.
3. Object - is an RDF resource that can be a URI or a literal[15] or a blank node.

2.3.1.4 Graphs

The core concept of the RDF is a set of triples, each containing a subject, predicate and object. A set of such triples is called an RDF graph. Figure 2.2 shows an example of a graph.

[13] http://www.w3.org/TR/rdf11-concepts/#section-IRIs - IRIs are a generalisation of Uniform Resource Identifiers (URIs).

[14] An RDF blank node is a node that does not contain any data or does not have a URI, but serves as a parent node to a grouping of data.

[15] A literal is a term used for values such as strings, numbers and dates

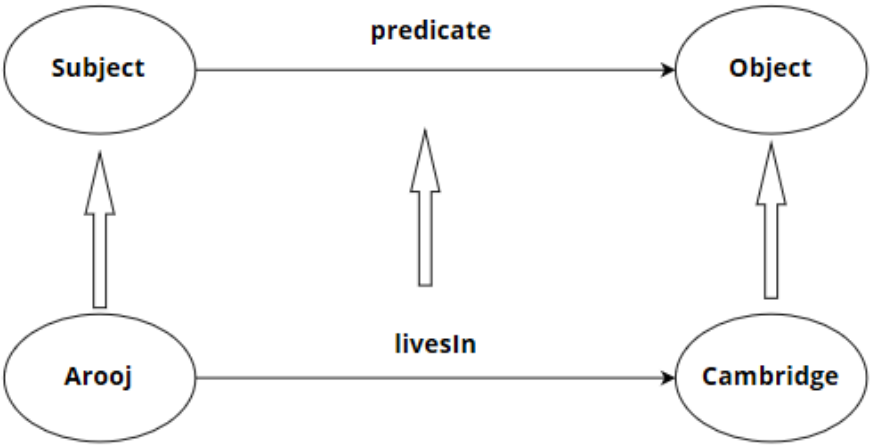


Figure 2.2: An example of an RDF graph (Source: Author, 2015)

2.3.2 Ontologies

Since ontologies have been used for different purposes in different disciplines, there are various definitions for an ontology. In computer science and knowledge engineering one commonly agreed definition for an ontology is given by Gruber [Gruber, 1993] who defines an ontology as 'An explicit formal specification of a conceptualization'. In simple language, ontologies can be defined as vocabularies to define meanings of data thus establishing a common understanding of terms between agents. When data is marked up with ontologies, it enables software agents to better understand the semantics and hence to relate linked data.

Ontologies are becoming of increasing importance in various fields such as intelligent systems, information integration, knowledge representation, information retrieval and knowledge engineering [Staab and Studer, 2013]. For an open system like the Semantic Web, different groups can define their ontologies in different ways e.g. the same ontology with different names. So, merely using ontologies does not ensure a common set of meanings. To solve this issue, some solutions have been introduced i.e. Ontology Matching [Faria *et al.*, 2014]. Ontology Matching is a technique that finds correspondences between semantically related entities of ontologies to solve the semantic heterogeneity problem. These correspondences can be used to merge ontologies and facilitate query answering on heterogenous ontologies [Shvaiko and Euzenat, 2013].

2.3.2.1 Ontology Representation

An ontology is comprised of four main components:

1. Concepts
   A Concept may be defined as a set or collection of objects. It may also be known as a class or a term. It is the fundamental element of a semantic domain. It represents a class or a group whose members share common properties. Like classes in objected oriented programming, the concepts can be represented in a hierarchical structure i.e. super class, child class. For example, a living thing can be a super class of 'Plants' and 'Animals' as child classes.

2. Instances
   An Instance may be defined as member of a class. It is also known as an individual. For example, *Ian* is an instance of 'Person' class.

3. Relations
   A Relation expresses link between two concepts in a given field. More specifically, it describes the relationship between the first concept, represented in the domain, and the second, represented in the range. For instance, teach could be represented as a relationship between the concepts 'Lecturer' and 'Subject'.

4. Axioms
   Axioms are used to validate the consistency of ontologies by imposing constraints on classes and instances. They are written using logic based languages.

2.3.2.2 Types of Ontologies

In the field of computer science, there are various classifications introduced for ontologies. [Van Heijst *et al.*, 1997] classified ontologies based on their purpose of usage i.e.

- Terminological ontologies - which specify terms in a knowledge base.
- Informational ontologies - that specify storage structure of data.
- Knowledge Modelling ontologies - which define concept models for knowledge.

[Guarino, 1998] defined ontology classification based on the level of generality i.e.

- Top-level ontologies - which define generic concepts independent of a domain e.g. time, event etc.
- Domain-level ontologies - that specify local concepts for a particular field e.g. Medicine, Education etc.

- Task-level ontologies - describe concepts related to a generic task e.g. buying, selling etc.
- Application-level ontologies - define concepts related to a particular task in a particular domain.

[Gómez-Pérez and Corcho, 2002] distinguish ontologies as

- Lightweight ontologies - these include concepts, taxonomies, relationships between concepts and concept properties.
- Heavyweight ontologies - that add axioms and constraints to lightweight ontologies.

2.3.3 Web Ontology Language (OWL)

OWL is a language to describe an ontology. OWL describes classes, properties and relations among conceptual objects in a way that facilitates machine interoperability of web content [Osman *et al.*, 2010]. OWL comes in three different dialects based on the expressivity of the language including OWL-Light, OWL-DL and OWL-Full (the last-named is the most expressive language) [Ye *et al.*, 2015].

2.3.4 Query Language

A query language allows users to formulate their queries in a simple way, without having any special proficiency in the technicalities of the underlying database [Schweikardt *et al.*, 2010]. There have been several languages proposed for querying RDF documents. Below is the list of commonly used query languages in chronological order.

2.3.4.1 RQL

RQL [Karvounarakis *et al.*, 2002] is a typed language following a functional approach. It relies on a formal graph model (as opposed to the triple based RDF query languages). RQL is defined by a set of basic queries and iterators which can be used to build new ones through functional composition [Karvounarakis *et al.*, 2003]. The key feature of the RQL is the smooth combination of schema and data query. However, RQL's semantics is not completely compatible with the RDF Semantics [Haase *et al.*, 2004]. Also, there are number of additional restrictions (such as having exactly one domain and range of a property) placed on RDF models that make RQL less compatible to query RDF.

2.3.4.2 SquishQL

SquishQL [Miller *et al.*, 2002] was designed to reflect graph structure. It uses SQL-like constructs to provide familiar structure to application developers. However, SquishQL faces a number of issues while evaluating a query i.e. (i) it is not possible to write queries that contain anonymous nodes and (ii) it is difficult to evaluate expressions in the absence of datatypes.

2.3.4.3 RDQL

RDQL[16] stands for RDF Data Query Language. It follows a SQL-like SELECT and WHERE pattern. RDQL is a safe language that offers support for datatypes. It does not interpret RDF Schema information and displays output as a table of variables [Bernstein and Kiefer, 2006].

2.3.4.4 SeRQL

SeRQL [Broekstra and Kampman, 2003] stands for Sesame RDF Query Language and is a querying and transformation language loosely based on RQL [Karvounarakis *et al.*, 2002] and RDQL [Bernstein and Kiefer, 2006]. SeRQL was primarily designed to reconcile ideas from the existing proposals to query RDF [Broekstra and Kampman, 2004]. SeRQL syntax is based on RQL but SeRQL's formal interpretation is based on the RDF Model Theory [Haase *et al.*, 2004]. SeRQL is not a safe language as it provides various recursive built-in functions [Haase *et al.*, 2004].

2.3.4.5 SPARQL

Since 2008, SPARQL has been the recommended semantic query language for RDF datasets by W3C [DuCharme, 2013]. Linked Open RDF Datasets provide a link to at least one SPARQL endpoint to directly query the dataset [Mehdi *et al.*, 2014]. The provision of SPARQL endpoints enable users (having knowledge of semantic technologies) to access semantic datasets using SPARQL queries. To facilitate the end users to retrieve information from LODs, there is a need for additional end user interfaces that support translation of natural language queries to SPARQL queries. SPARQL has been a core focus of research and development communities for Semantic Web technologies since 2008, with various research proposals, benchmarks, open-source and commercial tools introduced to address the challenges of processing SPARQL [Buil-Aranda *et al.*, 2013]. Since

[16]RDQL - a query language for RDF (https://www.w3.org/Submission/RDQL/).

SPARQL is a W3C recommendation and up-to-date with RDF semantics, the other query languages will not be considered further in this thesis.

2.4 Semantic Search

The WWW has become an essential part of everyday life. It has changed the face of business and communication. Typical uses of WWW are making use of information, searching for and getting in touch with people, searching or ordering products online.

The concept of a web search has evolved with the growing volume of data resources. In the past, various techniques have been used to store data and meta-data so that the data can be found efficiently. For a web search, the widely used technique is SEO (Search Engine Optimisation). Since the information retrieval is syntax based, it has a number of limitations e.g. low precision and high recall.

A semantic search promises to eliminate issues that arise from syntax based search. For the Semantic Web, a semantic search requires the search system to understand the semantics of a user query and process it to generate a relevant SPARQL query. To develop a system that supports query creation is far more complex than a keyword based search approach. Currently, a large part of semantic web research deals with the creation of queries for semantic data because writing search terms in a query language like SPARQL is very difficult especially for end users.

2.4.1 Types of Semantic Search Queries Based on Input Category

There are three main types of generic end user search queries [Jansen *et al.*, 2008]

1. Navigational search queries - where the user aims to find the Uniform Resource Locator (URL) for a web page
2. Transactional search queries - where the user targets some transaction such as buying something online
3. Informational search queries - where the user intends to find some information based on certain criteria

This thesis focuses on informational search queries to find information from linked open datasets since the datasets are the only available open source option and they are best

suited for informational queries. There are a number of ways semantic search queries can be asked, these include:

1. **Formal Queries**

   Formal queries are written in a formal query language e.g. SPARQL. Most of the linked open datasets provide a SPARQL end-point accessible to users and agents to retrieve semantic data using formal queries.

2. **Natural Language Queries**

   These types of queries are written using a natural language like English. Natural language queries are the most user friendly type of queries and at the same time the most complicated ones to implement.

3. **Keyword-based Queries**

   This is where a set of keywords to identify user interest is used to formulate a keywords-based query e.g. 'Anglia Ruskin Cambridge'

4. **Form-based Queries**

   Form-based queries provide input fields i.e. text fields, drop downs, check boxes or radio buttons that help the users to customise their search options for a particular domain or a specific area of search.

2.5 Challenges for a Semantic Search

The Semantic Web faces a number of challenges regarding searching data from RDF datasets, these include:

2.5.1 Knowledge of Datasets

To query linked open datasets, the user has to be familiar with the structure of the datasets. The user also needs to know the relationship between entities. The user has to express the relationships defined between concepts in the RDF triple patterns, which involves browsing RDF datasets. The task to express relationship gets more complicated when utilising federated queries[17], which even in trivial scenarios implies browsing at least two to three datasets [Jain *et al.*, 2010].

[17]Federated query is the ability to take a query and provide solutions based on information from many different sources (https://www.w3.org/2009/sparql/wiki/Feature:BasicFederatedQuery).

2.5.2 Schema Heterogeneity

Schema heterogeneity (also known as Semantic heterogeneity) results from different resources using different vocabularies to define similar concepts or resources [De Virgilio *et al.*, 2012]. An ontology for a dataset may contain a number of namespaces[18] to use various schemas[19]. Say, there are three namespaces that define schema for the concept 'Person' (as given below)

http://namespace1.url/Person
http://namespace2.url/Person
http://namespace3.url/concepts#Person.

If a user wants to search the names for persons, he needs to know all the schemas that define a person as well as the datasets which use these schemas. This issue makes LOD datasets less accessible to the end users.

2.5.3 Result Optimisation

An end user, who interacts with an application to perform queries on LOD datasets, needs to be presented with results in a user-friendly way [Ell *et al.*, 2011]. The Semantic Web identifies entities and their relations by Uniform Resource Identifiers (URIs). Often the LOD datasets return results (as set of URIs) in XML[20], JSON[21] or plain HTML[22] formats that are not very readable as compared to the existing user friendly search interfaces (e.g. Google, Bing). An effective end user interface should optimise results for better readability and visualisation. [Ell *et al.*, 2011] suggest using rdfs:label and rdfs:comment from the RDF vocabulary to provide a human-readable version of a resource's name besides its URI. However, for many URIs the human readable labels are not defined.

2.5.4 Query Optimisation

Although the semantic search aims to retrieve relevant results, it can still have unwanted results. For example the word 'owl' can bring results for the animal 'owl' or the ontology language 'OWL'. In such a scenario, the search engine needs to confirm from the user

[18] A namespace defines scope of an element e.g. FOAF is a namespace that defines schemas for Person, Organisation etc.
[19] A schema (in this context) is a representation of a concept such as Book, Country
[20] Extensible Markup Language
[21] JavaScript Object Notation
[22] Hypertext Markup Language

the exact query he is looking for. Existing non-semantic search engines solve this issue by auto-suggest options. For a semantic search system, query optimisation is a challenge as some complex queries can have a number of interpretations. A similar approach was adopted by [Stojanovic, 2003] for their Librarian Agent to find the ambiguities in a user request.

2.5.5 Result Ranking

A search query can interrogate multiple datasets and return various results. This scenario raises the question of how the results should be sorted. The conventional search engines use various techniques to rank results i.e. relevancy, popularity or search engine's own ranking tables [Beel and Gipp, 2009]. Since the core data structure of the semantic web is based on its underlying ontologies, the result ranking for semantic search should be ontology based. This dependency further increases the challenge of result ranking since it needs to deal with the heterogeneity problem of ontologies.

2.5.6 Heterogeneous Ontologies

Ontologies provide a medium to identify the meanings of data and may allow some reasoning on the data. Unfortunately, different viewpoints lead ontology designers to produce the same concepts differently e.g. using various terms for the same concept or the same concept defined by multiple namespaces. This results in heterogeneous ontologies for the same domain of interest. There can be a number of solutions to solve this issue. The simplest solution is that all communication systems agree on using same set of ontologies. However this situation is not always possible or feasible. Another way of solving the heterogeneity problem is to use a single ontology that has all terms used in every application but adding new terms or meanings to such an ontology is nearly impossible. These issues call for the ontology mapping techniques. The core challenge to map heterogenous ontologies is the scale of the search space [Zhang *et al.*, 2015]. The most reliable technique for ontology mapping is to do it manually but this is frequently infeasible when the size of ontologies is large or steadily increasing.

2.6 Do we need a separate semantic search engine?

End users do not always search on a particular question. Instead, they may want to find everything related to a word or set of words. Text based searches are prone to ambiguities and errors but sometimes the user actually wants a text based search (for example to

match all the documents containing a particular piece of text). Modern search engines like Google[23] and Bing[24] have a number of rich features to support syntax based searches e.g. keywords tagging, auto-correct of grammar and spelling, auto-suggest queries and identifying the query type. Retaining these rich features and adding a model layer that can process information queries on LOD datasets can help to provide a complete search experience while reducing the effort in building a whole new system. This research looks into the possibilities of building such a system.

2.7 Summary

This chapter has explained the background including semantic web architecture and its technologies (e.g. RDF, ontologies, SPARQL and OWL). The author has also discussed the concept of semantic search, different types of semantic search queries followed by a discussion to see if a separate semantic search engine is needed. From the background review of the semantic search (in this chapter), the author believes that the limitations of a semantic search can be handled by a search tool that supports natural language user queries and can be integrated with other search systems. The next chapter explains the methodology and conceptual framework for this research.

[23]https://www.google.co.uk - Google Search Engine
[24]https://www.bing.com/ - Bing Search Engine

Chapter 3

Methodology

3.1 Introduction

Research Methodology is a systematic way to solve a problem and it is necessary for a researcher to know the research methods as well as the research methodology [Kothari, 2004]. The objective of this chapter is to demonstrate how the research will be conducted for the current work.

Section 3.2 analyses current research approaches. Section 3.3 introduces the research methodology chosen by the author and explains its adoption for the research execution. Section 3.4 describes the frameworks for Design Research detailing the methods used. Section 3.5 presents the design plan for the current research and explains the application of a Design Research Methodology to the execution of this work.

3.2 Current Research Approaches

A number of research methodologies are available, but [Oates, 2006] defines a generic research process for a multidisciplinary field like Information Systems. Figure 3.1 shows Oates's model that describes different research lines based on different research methods. The steps in a research process are not mutually exclusive, they need not necessarily follow each other in a specific manner and the researcher may need to anticipate each step in the process [Kothari, 2004]. For this work the author has chosen the Design Research methodology as it has greatest relevance to the specialities of this study. The dominant research paradigms for Information Systems are mostly borrowed from the social and natural sciences. Recently, interpretive research paradigms have been accepted, but the resulting research output is still mostly explanatory and, it could not be often applicable

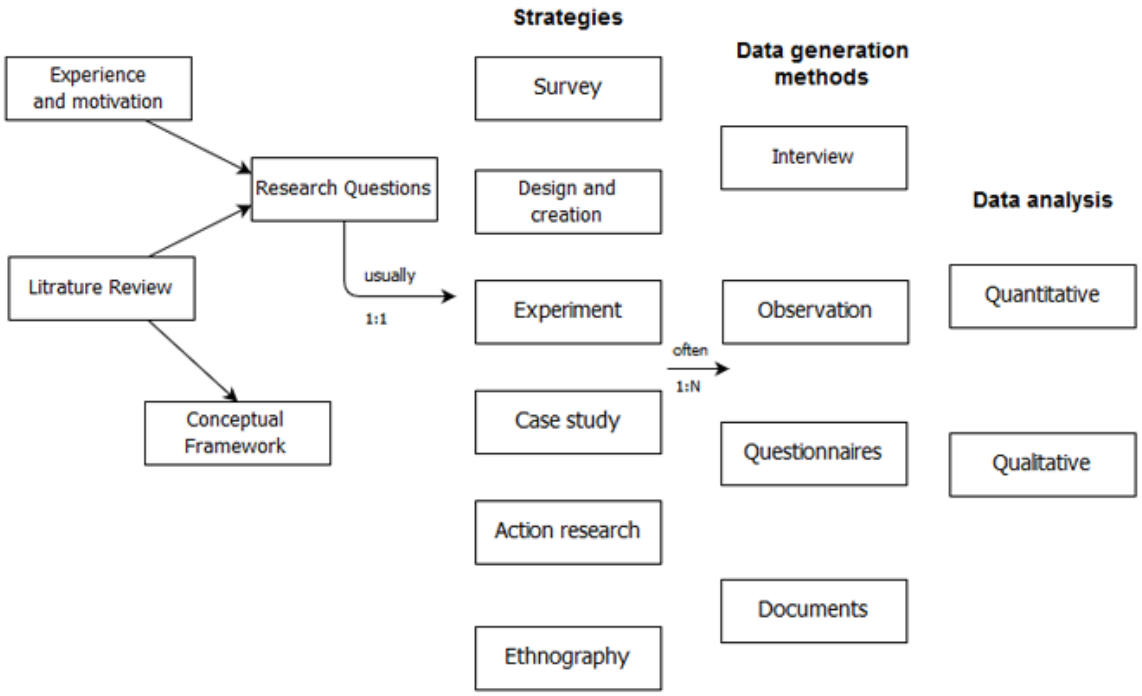


Figure 3.1: Model of a generic research process (Source: [Oates, 2006])

to the solution of problems encountered in research and practice [Peffers *et al.*, 2007]. Design research is motivated by the desire to improve the environment by the introduction of innovative artifacts and the processes for building these artifacts [Hevner, 2007].

3.3 Design Research

"It is clear from the investigation of numerous engineering failures and disasters that the simplest and most fundamental design methods, guidelines, rules and recommendations are still not understood, accepted, or used" [Hales *et al.*, 2003]. Design Research aims to fill the gap for those missing methods. Design in this context, refers to the activities required to complete a task or produce a product idea. Design Research can be divided in two categories Innovative Design Research and Routine Design Research. If the knowledge to create an artifact already exists, then the design research is *routine* otherwise it is *innovative* [Vaishnavi *et al.*, 2013]. This thesis is based on Routine Design Research category.

The selection of Design Research for this research is based on its overall aim, that is, to make design more efficient and effective. [Blessing and Chakrabarti, 2009] define the objectives of design research as follows:

- The formulation and validation of theories

- The development and validation of the design support founded on theory

3.4 Design Research Methodology

[Simon, 1996] introduced the concept of an artefact as a link between internal and external environments that attains the goals of a satisfied design. Later, [Owen, 1998] presented a general model to understand design disciplines and design research process: knowledge is generated through action. Consequently, [Vaishnavi and Kuechler, 2004] identified design as a process and clearly defined the steps involved. Design Research could not get much recognition at the start because it lacked a methodology to serve as a commonly accepted framework and a template for its presentation [Peffers *et al.*, 2007].

[Blessing and Chakrabarti, 2009] define Design Research Methodology as a framework that helps develop and validate knowledge systematically, at the same time ensuring that the research is scientific and delivers valid results. There are multiple approaches adapted for Design Research. [Duffy and O'Donnell, 1998] introduced a general research methodology (as shown in Figure 3.2) based on the Design Research framework proposed by [Duffy and Andreasen, 1995]. This model suggests that the research problem should be driven from the literature and design in practice. However they did not explain their methodology in detail. [Antonsson, 1987] and [Eckert *et al.*, 2003] have also proposed models based on design research. The common element in these models, is an emphasis on hypothesis.

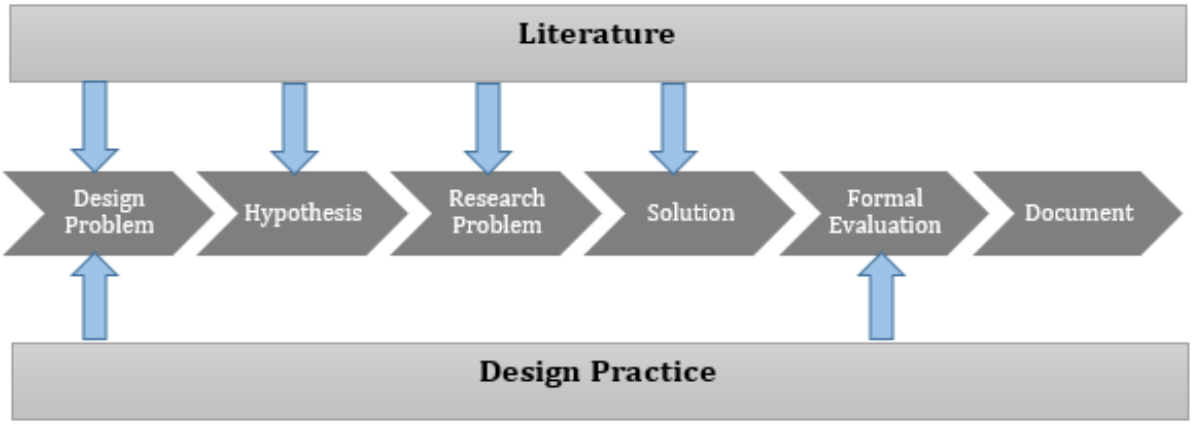


Figure 3.2: DRM Model by [Duffy and O'Donnell, 1998] (Source: Author, 2015)

Design Research Methodology introduced by [Blessing and Chakrabarti, 2009] proposes that the Design Research should employ a generic framework (as shown in Figure 3.3) where they define four stages of a research process.

1. Research Clarification Stage
2. Descriptive Study I
3. Prescriptive Study
4. Descriptive Study II

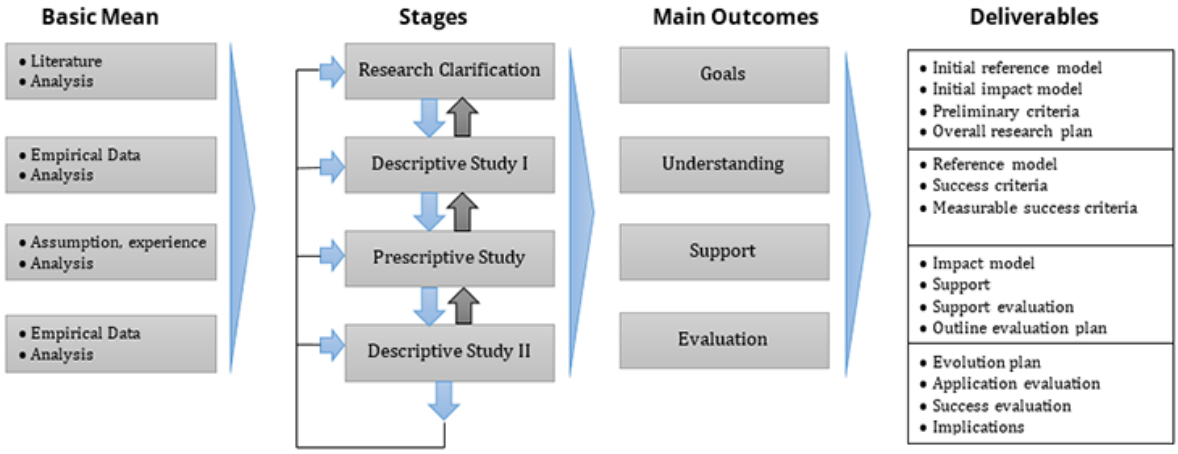


Figure 3.3: DRM Model (Source: [Blessing and Chakrabarti, 2009])

[Blessing and Chakrabarti, 2009] defined seven types of design research based on the type of study required at different stages in their DRM framework (as shown in Table 3.1).

No.	Research Clarification	Descriptive Study I	Prescriptive Study	Descriptive Study II
1.	Review based	Comprehensive		
2.	Review based	Comprehensive	Initial	
3.	Review based	Review based	Comprehensive	Initial
4.	Review based	Comprehensive	Review based Initial/Comprehensive	Comprehensive
5.	Review based	Comprehensive	Comprehensive	Initial
6.	Review based	Comprehensive	Comprehensive	Comprehensive
7.	Review based	Comprehensive	Comprehensive	Comprehensive

Table 3.1: Types of projects in DRM (Source: [Blessing and Chakrabarti, 2009])

3.5 Implementation of DRM for this research

The author adopted the third research type from Table 3.1 as follows.

1. In the *Research Clarification* stage, the author undertook an intensive literature review, discussions with supervisors and the research communities in seminars and

through multiple platforms like LinkedIn[25], Research Gate[26]. This stage has been the foundation to define the research goal and overall research plan (Chapter 1).

2. In the *Descriptive Study I* stage, the literature on semantic search and related Semantic Web technologies was reviewed to achieve an increased understanding of the problem domain (Chapter 2).

3. In the *Prescriptive Study* stage, the author used the understanding attained from literature review (from the previous stages) to evaluate the tools available in the semantic community to support semantic search (Chapter 4). The evaluation of related tools helped the author to identify the criteria to be met by semantic search tools. The criteria was formulated into a proposed framework (Chapter 5). Individual modules of the framework will be detailed in Chapter 6, 7, 8 and 9.

4. In the *Descriptive Study II* stage, an initial evaluation of the developed framework was carried out by developing a prototype based on the proposed model (Chapter 10). The researcher used British Library free datasets and SPARQL end-point as a test-bed for initial evaluation. The method was then applied to some other datasets such as DBPedia.

Figure 3.4 maps the thesis chapters against the DRM model. Chapter 1 covers the clarification stage and identifies the research goals from literature review. At this stage, the author short-lists the research questions to be explored at later stages. The *Descriptive Study I* stage (covered in Chapters 2 and 3), explores the background of the semantic data technologies and pinpoints the challenges faced by semantic search. The author focuses more on the *Prescriptive Study* stage (covered in Chapters 4, 5, 6, 7, 8 and 9) to find out the limitations of existing semantic search systems and consequently proposes a framework as a solution to facilitate semantic search. Finally, the *Descriptive Study II* stage (covered in Chapter 10) describes the initial evaluation of the developed framework followed by Chapter 11 to sum up the conclusions.

[25] http://linkedin.com - LinkedIn is a business-oriented social networking service.
[26] http://www.researchgate.net - ResearchGate is a social networking site for scientists and researchers to share their research ideas.

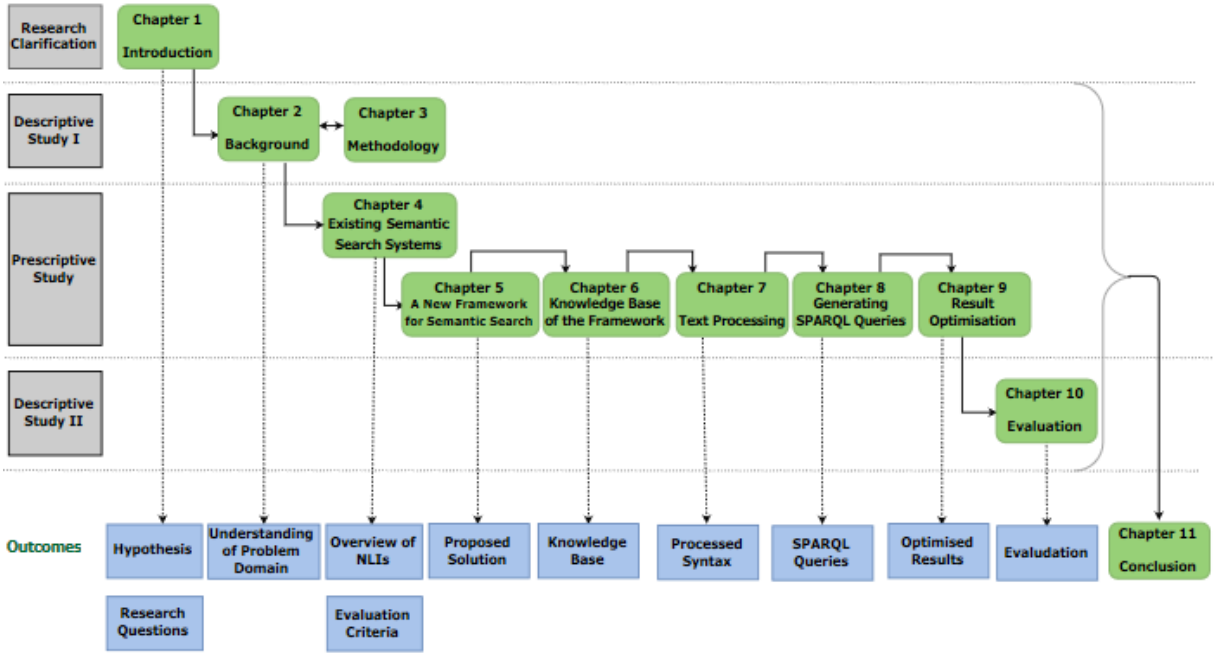


Figure 3.4: Comprehensive thesis structure in context of the adopted methodology (Source: Author, 2015)

3.6 Summary

This chapter has explained the methodology adopted for this research work. [Blessing and Chakrabarti, 2009] define seven different design research approaches that can be adopted based on project type. The author has chosen the third type of methodology that involves four stages i.e. (i) Research Clarification, (ii) Descriptive Study I, (iii) Prescriptive Study, and (iv) Descriptive Study II.

At the first stage of the methodology, the author identified the research goals from literature review and peer discussions. *Descriptive Study I* stage identified the shortcomings of existing semantic search tools which was followed by the *Prescriptive Study* stage in terms of the proposed framework and its prototype embodiment. Finally, the *Descriptive Study II* stage involved initial evaluation of the developed framework.

The next chapter utilises the literature review from *Descriptive Study I* stage in Chapter 2 and undertakes *Prescriptive Study* to analyse existing semantic search tools.

Chapter 4

Existing Semantic Search Systems

4.1 Introduction

Semantic search promises to provide more relevant and accurate results as compared to the syntax based search. However, in order to utilise semantic search, end users need to master the complexity of its query languages and familiarity with the underlying ontolgies [Zhou *et al.*, 2007]. The semantic search approaches are characterised by their high level of diversity both in their features as well as their capabilities [Elbedweihy *et al.*, 2012a]. [Uren *et al.*, 2007] categorised informal semantic search systems in four categories i.e. (i) keywords-based, (ii) form-based, (iii) view-based and (iv) natural language based systems.

Later [Wei *et al.*, 2008] classified semantic search systems into six categories based on their methodologies and scope. They did not limit their investigation to certain criteria instead they undertook a generic approach and presented a framework for semantic search comprising of six components i.e. semantic data acquisition, knowledge acquisition, data integration and consolidation, semantic search mechanisms, semantic search services, and result presentation. Their framework lacks details of the components and it does not say anything about the query interface. Similarly, the search interface evaluation conducted by [Elbedweihy *et al.*, 2012a] showed that the Natural Language Interfaces (NLIs) are the most acceptable by end-users (who prefer NLIs over keyword matching, menu-based or graphical interfaces). Investigations by [Danica and Bontcheva, 2009] and [Kaufmann and Bernstein, 2010] on usefulness and usability of NLIs have shown that end users need a similar means of query access (as provided by conventional search interfaces) and NLIs are believed to be a practical solution.

This chapter looks into existing systems that support controlled or full Natural Language (NL) queries to access linked open datasets. It is not the author's intention to cover

all existing approaches, but to review the state-of-the-art to investigate the capabilities of different systems with respect to their approach, scope, capacity of query translation, natural language support etc. Section 4.2 provides an overview of the natural language based semantic search systems. Section 4.3 provides an analysis of the available NL based systems and identifies the NLIs with maximum features. Section 4.4 further compares the short listed NLIs from Section 4.3. The chapter finishes with discussion (Section 4.5) and summary (Section 4.6).

4.2 Natural Language Interfaces for Semantic Search

For the success of the Semantic Web, it needs to be easily accessible to the end users via interfaces that facilitate end users to browse and query semantic datasets. There are numerous tools that perform one or more of these tasks. This section attempts to evaluate existing NLIs for the Semantic Web. One of the very first efforts to evaluate NLIs for semantic search was made by Kauffman and Bernstein who mainly investigated usability of four NLIs i.e. Ginseng, NLP-Reduce, Querix, and Semantic Crystal [Kaufmann and Bernstein, 2007]. Kaufmann and Bernstein's usability tests with 48 users showed that users prefer full-sentence query to keywords. [Elbedweihy *et al.*, 2012a] evaluated Ginseng, NLP-Reduce, K-Search and PowerAqua NLIs based on performance (i.e. speed of execution), usability (with different input styles) and analyses of results returned. Their work was focused on the usability of NLIs and their results showed that most of the users were not able to understand the results presented to them because of the technical jargons used.

In this section the author analyses the semantic search tools that provide natural language interfaces to access semantic data. At the time of writing, there are a number of tools available that support NLIs for semantic search. [Sharef and Noah, 2012] divide semantic search NLIs into three categories i.e.

1. Form-based and Graph-based e.g. NLPReduce [Kaufmann *et al.*, 2007], ORAKEL [Cimiano *et al.*, 2008] and Semantic Crystal [Kaufmann and Bernstein, 2007]).

2. Controlled Natural Language support e.g. Querix [Kaufmann *et al.*, 2006], GINO [Bernstein and Kaufmann, 2006] and Ginseng [Bernstein *et al.*, 2005]).

3. Full Natural Language support e.g. QuestIO [Damljanovic *et al.*, 2008] and its upgraded version FREyA [Damljanovic *et al.*, 2011], AquaLog [Lopez *et al.*, 2005] and its revised version PowerAqua [Lopez *et al.*, 2011]).

The author only focuses on the tools which have support for controlled or full natural language queries. In the following sections, the author reviews these existing NLIs in chronological order.

4.2.1 AquaLog

AquaLog [Lopez *et al.*, 2005] uses General Architecture for Text Engineering (GATE) [Cunningham, 2002] to parse user queries and find word forms i.e. nouns, verbs and question identifiers. AquaLog identifies classes and relationships by mapping terms in the ontology and synonyms obtained from WordNET[27] [Miller, 1995].

AquaLog uses ontological reasoning to learn user's search behavior to improve user experience over time, but this learning mechanism is confined to a particular user. AquaLog supports 23 types of queries and does not answer a query that is not classified in these 23 types as this system heavily relies on its controlled language [Danica and Bontcheva, 2009]. Its architecture is shown in Figure 4.1.

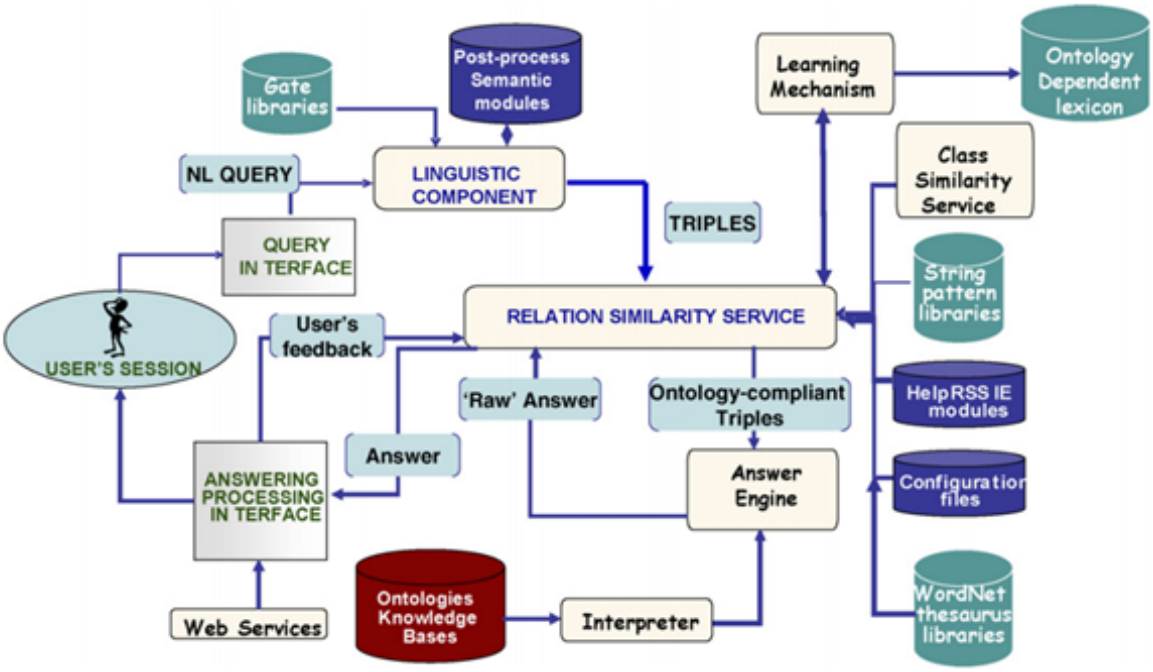


Figure 4.1: Architecture of AquaLog (Source: [Lopez *et al.*, 2007])

[27] WordNet is a large lexical database of English which groups nouns, verbs, adjectives and adverbs into sets of cognitive synonyms (synsets).

4.2.2 Ginseng

Ginseng [Bernstein *et al.*, 2005] provides natural language (NL) querying access to semantic ontologies. Ginseng only searches the vocabulary defined by its loaded ontologies. It also does not try to interpret queries. This feature limits user possibilities, especially if the user aims to search for some particular instances in a knowledge base.

4.2.3 Querix

Querix [Kaufmann *et al.*, 2006] accepts full English questions starting with question identifiers e.g. what, when, how, does etc. Querix is a closed domain system that describes only a single domain. Querix translates natural language queries to SPARQL language. Querix supports ontology-based question answering. It does not solve natural language ambiguities but it uses clarification dialogs in case of ambiguities. There is no information currently available on Querix architecture.

4.2.4 GINO

GINO (A *G*uided *I*nput *N*atural Language *O*ntology Editor) [Bernstein and Kaufmann, 2006] is an extension of Ginseng that provides controlled natural language query access to OWL knowledge base. GINO's architecture is based on simple sentence grammar and it uses Jena[28] Ontology model and SPARQL Engine. GINO is not a full-fledged ontology editing tool and its use of controlled language minimises the expressivity of a user query. Its architecture is shown in Figure 4.2.

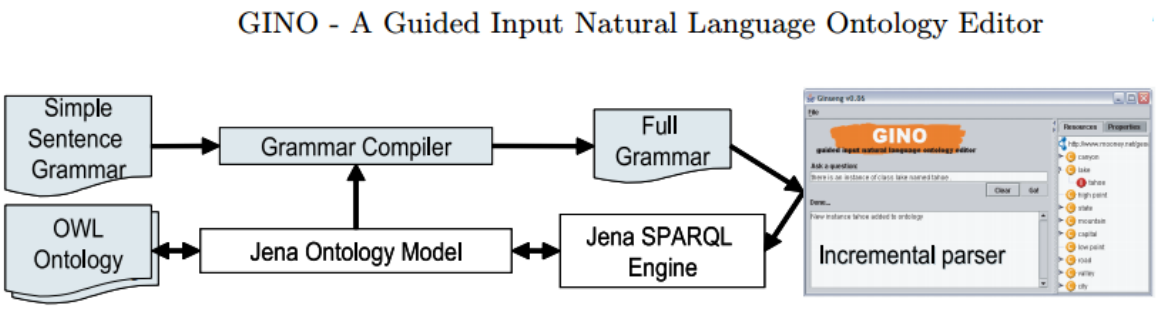


Figure 4.2: GINO Architecture (Source: [Bernstein and Kaufmann, 2006])

[28] A free and open source Java framework for building Semantic Web and Linked Data applications.

4.2.5 e-Librarian

e-Librarian [Linckels and Meinel, 2007] is a closed domain system that splits words into triples of the form (subject, object, verb). e-Librarian accepts a complete question and returns a set of documents. For ontology mapping, it uses a domain specific dictionary. Though the use of a limited dictionary improves system performance, however the dictionary needs to be built manually. Its architecture is shown in Figure 4.3.

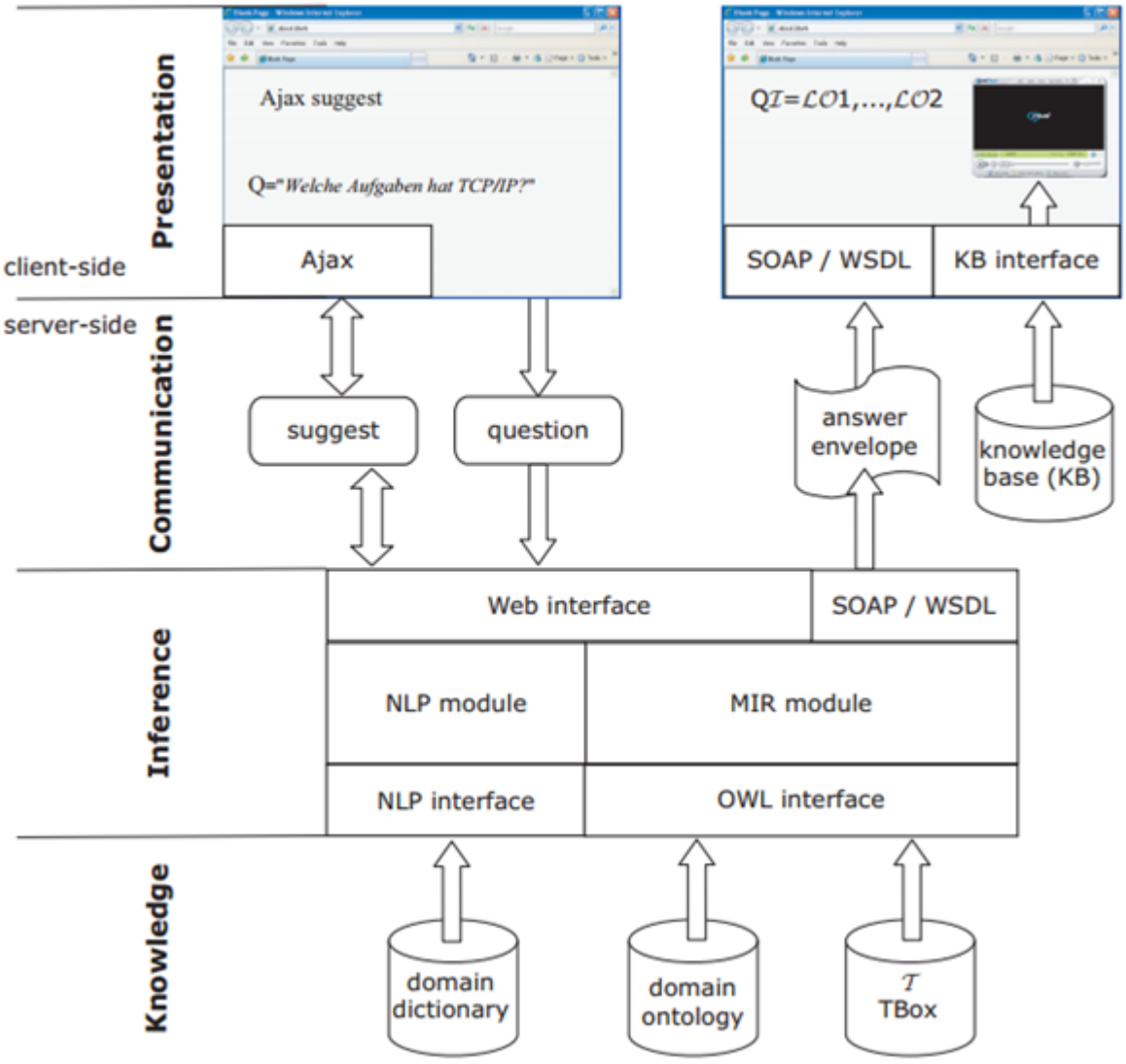


Figure 4.3: Architecture of e-Librarian (Source: [Linckels, 2008])

4.2.6 QuestIO

QuestIO [Damljanovic *et al.*, 2008] is a text-based query interface to OWL ontologies. QuestIO supports full NL queries and translates user queries to SeRQL [Broekstra and

Kampman, 2003] queries. No details of its architecture are available.

4.2.7 PowerAqua

PowerAqua [Lopez *et al.*, 2011] is a successor of AquaLog. Unlike Aqualog, it does not map terms to ontology triples instead it tries to match the query to the ontology that covers most of the terms within that query. PowerAqua handles various ontologies across the web. Its architecture is shown in Figure 4.4.

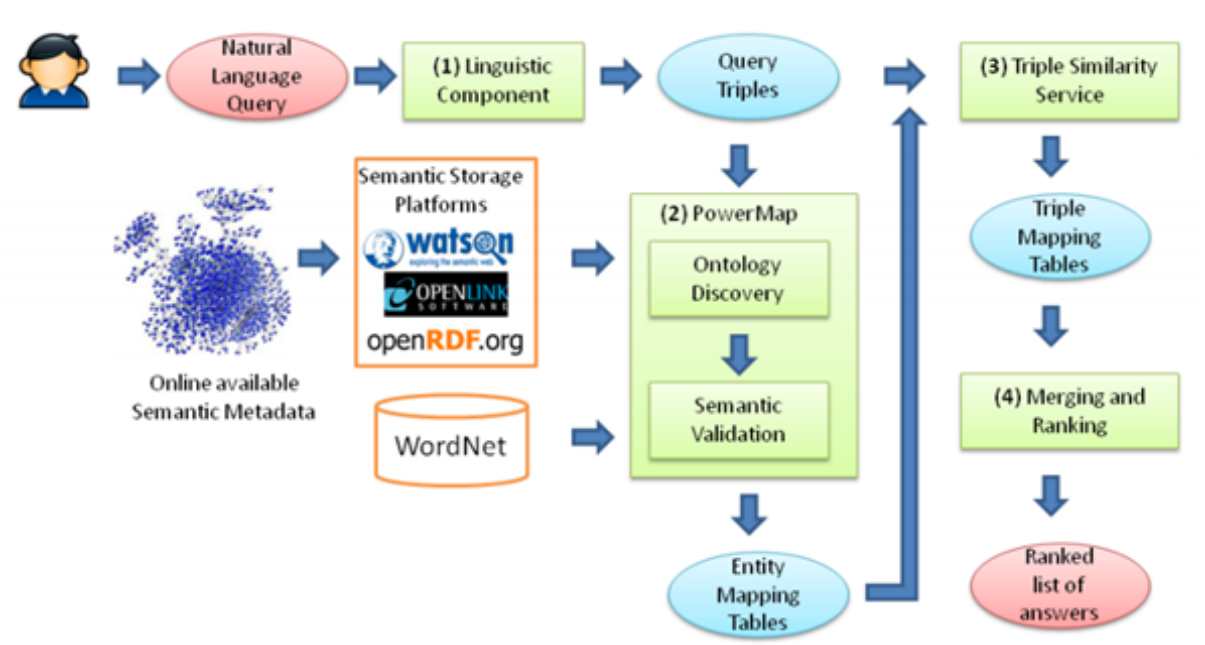


Figure 4.4: Architecture of PowerAqua (Source: [Lopez *et al.*, 2011])

4.2.8 FREyA

FREyA (named after *F*eedback, *R*efinement and *E*xtended Vocabular*y* *A*ggregation) [Damljanovic *et al.*, 2011] aims to resolve syntax ambiguity in user queries by auto-suggest functionality using clarification dialogs. It guides the user to select the exact semantics for the ambiguous terms unless it finds a non-ambiguous query that is recognised by the system. FREyA will not give any results unless the ambiguities are resolved.

FREyA is an open domain system that has a similar approach to Querix and uses clarification dialogs. FREyA learns user's search behaviour and re-uses this learning for all other users [Damljanovic *et al.*, 2010]. Its work-flow is shown in Figure 4.5.

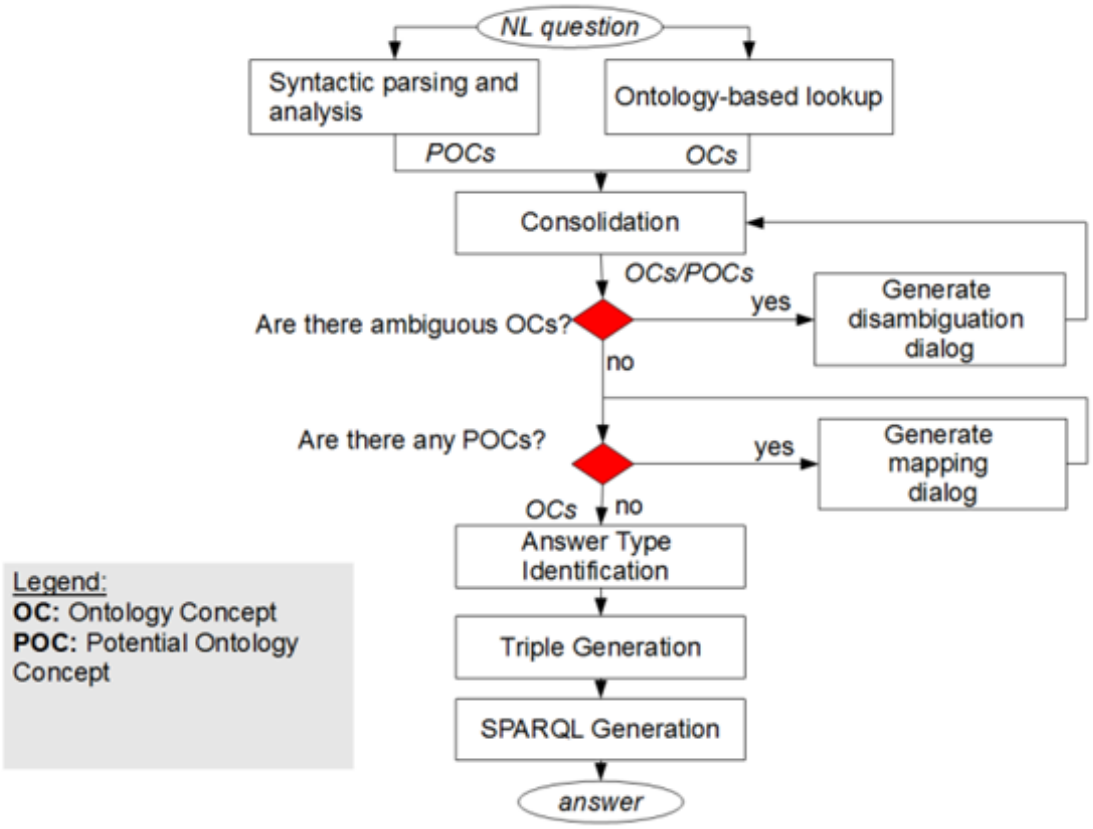


Figure 4.5: Work-flow of FREyA (Source: [Damljanovic *et al.*, 2011])

4.2.9 QAKiS

QAKiS [Cabrio *et al.*, 2012] uses a Wiki Framework and relational patterns extracted from Wikipedia[29]. In the first step, it finds EAT (Expected Answer Type) based on predefined rules. QAKiS performs searches on DBPedia[30] using the technique of named-entities[31]. QAKiS has a limitation in that it needs at least one named entity in the query. QAKiS is also not scalable to other domains. Its architecture is shown in Figure 4.6.

[29]http://wikipedia.org - WikiPedia is a free encyclopedia built collaboratively using wiki software.

[30]http://dbpedia.org - DBpedia is a project aiming to extract structured content from the information created as part of the Wikipedia project.

[31]A *named entity* in this context is an item that is identifiable in a set of items e.g. names, locations etc.

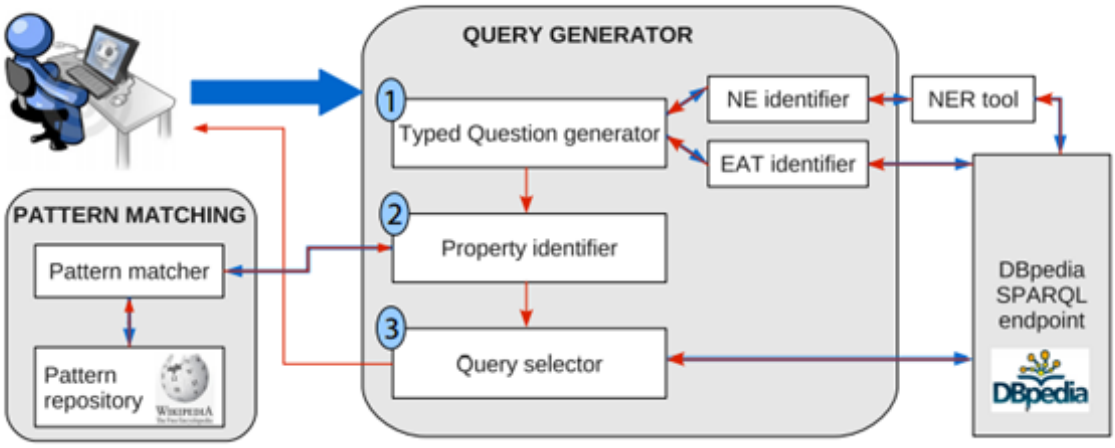


Figure 4.6: Architecture of QAKiS (Source: [Cabrio *et al.*, 2012])

4.3 Analysis of Existing Semantic Search Systems

End users expect a search tool to support natural language queries [Tablan *et al.*, 2008] and understand their required results by identifying the type of query and expected result category [Tyckoson and Dove, 2014]. A natural language interface facilitates end users in typing natural language queries without exposing the underlying structure of a system. Although there are a number of Natural Language Interfaces available, at the moment they are limited in their ability to process query containing negation, conjunction, quantification, temporal and numeric expressions [Sharef and Noah, 2012]. Some of these systems (like FREyA) force the user to answer multiple queries to solve simple queries such as "which cities are located in Europe?" or "where is Europe?" (see Figure 4.7). Usability tests for semantic search NLIs by [Kaufmann and Bernstein, 2007] and [Elbedweihy *et al.*, 2012a] showed that the results were not readable by the users due to poor presentation.

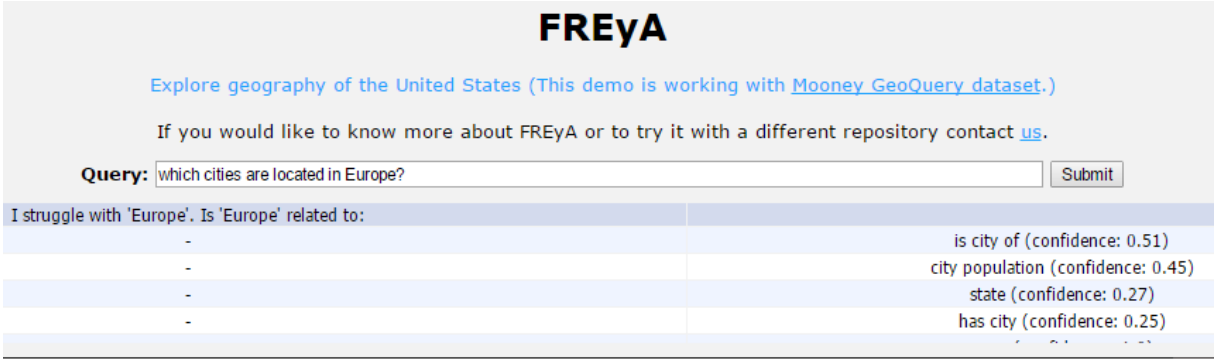


Figure 4.7: Screenshot of Testing on FREyA (Source: Author, 2015)

To investigate the problems with existing NLIs (that cause their limitations), the author has chosen the systems with richer features. Table 4.1 shows a quick overview of existing NLIs considering the level of natural language support, scope of domain and ability to translate a user query to a SPARQL query.

From this comparison of existing NLIs (as shown in Table 4.1), the author identified three systems (PowerAqua, FREyA and QAKiS) that fulfil the criteria of interest i.e. the systems that are open domain and support full natural language queries as well as conversion of user queries to SPARQL queries. The shading (in Table 4.1) shows systems with limitations, not considered further.

Search Systems	NL Support	Scope	Query Translation
AquaLog	Full	Closed	-
Ginseng	Controlled	Closed	Y
Querix	Controlled	Closed	Y
GINO	Controlled	Closed	Y
e-Librarian	-	Closed	Y
QuestIO	Full	-	-
PowerAqua	Full	Open	Y (Watson API)
FREyA	Full	Open	Y (Watson API)
QAKiS	Full	Open	Y

Table 4.1: A comparison of existing semantic search NLIs (Source: Author, 2015)

4.4 Comparison of the Selected NLIs

Existing semantic search NLIs (such as PowerAqua, FREyA and QAKiS) have been reviewed and evaluated by a number of researchers. [Sharef and Noah, 2012] evaluated FREyA against their AutoSPARQL and concluded that FREyA's learning component has not shown significant performance. [Elbedweihy *et al.*, 2012b] evaluated PowerAqua and FREyA and reported a relation matching problem with PowerAqua. They also found that both tools could not manage to return all results.

Since the interest of this work is to design a system that can work on its own as well as integrated with other search systems, the author compares the chosen systems based on their capacity of accessibility, portability, extensibility, interoperability and result readability. The comparison is based on literature review and some tests on the available demo test points. Since no demo test point could be found for PowerAqua, the available literature and video demonstration[32] have been used for this stage of study.

[32]http://technologies.kmi.open.ac.uk/poweraqua/demo.html - Link for PowerAqua demo video.

4.4.1 Accessibility

Accessibility in this context, means the measure of convenience with which end users or software agents can access LOD datasets. RDF datasets are accessible through dereferenceable HTTP URIs or SPARQL endpoints [Cheung *et al.*, 2009]. This requires the end users to memorise HTTP URIs or a command over SPARQL query language to access semantic data. To facilitate end users and other search tools to access semantic data, a semantic search tool needs to be able to translate natural language queries to SPARQL queries [Freitas and Curry, 2014]. The available NLIs support SPARQL query translation though they are not able to process complex queries. QAKiS deals with very basic query sets only [Usbeck *et al.*, 2015]. FREyA limits the expressivity of a query by using clarification dialogues. PowerAqua performs query conversion based matching between words of the user query and concept relations of a triple store. However, a word-based match may fail to detect the relevant context hence producing a wrong match [Cabrio *et al.*, 2012].

4.4.2 Portability

Portability in this context, can be defined as the possibility of applying a system to a new dataset [Ferré, 2014]. A portable semantic search system should be able to move between ontologies without any need for domain-specific reconfiguration [Uren *et al.*, 2007]. PowerAqua is a newer version of AquaLog and it retains the portability feature of the former [Lopez *et al.*, 2006]. FREyA and QAKiS are ctegorised as portable systems per their documentations.

4.4.3 Extensibility

The extensibility enabled by semantic technologies allows the consumption of data from many different domains [Patton *et al.*, 2014]. Extensibility of a semantic search system can be defined as the ability to accommodate new domains whilst not significantly modifying underlying architecture of the system. The author differentiates extensibility from portability such that the portability is the ability of a system to work on a different underlying structure (i.e. different domain ontologies) whereas the extensibility is the capacity of a system to merge more than one domains. To the best of author's knowledge, none of the existing semantic search systems cater for the feature of extensibility. FREyA, PowerAqua and QAKiS are portable systems but there is no explicit claim for extensibility.

4.4.4 Interoperability

[Desourdis, 2009, p. 4] defines interoperability as "the ability of two or more systems or components to exchange information and to use the information exchanged". One of the approaches to interoperability is the use of open APIs [Petcu, 2011]. The author attempts to design an API based system for semantic search that is integrable to existing web search systems. To enable a system to be integrated with another system, the former should have a very simple interface to accept a request and send a proper response back. A system like FREyA (using clarification dialogues) can be best used in a user interactive mode but it may not be suitable in a system to system communication. The author proposes a very basic integration where a request will be sent as a natural language query and response will be returned in JSON[33] format with some extra fields to identify multiple interpretations of the user query.

4.4.5 Result Optimisation

Among various challenges for creating an efficient semantic search tool is the question of how to optimise query results. Result optimisation in a view of user friendliness involves result ranking and result layout for better visualisation of those results. QAKiS (Figure 4.8) is the only system that provides good presentation of results but QAKiS matches only one relation per query and it relies on basic questioning [Usbeck *et al.*, 2015].

[33] JavaScript Object Notation - http://json.org/

Figure 4.8: Screenshot of Testing on QAKiS (Source: Author, 2015)

4.5 Discussion

An analysis of semantic search NLIs (in Section 4.3) showed that each system has its features as well as limitations. All existing systems use different architectures and different approaches for query translation but none of the systems completely fits into author's required criteria (as described in Section 4.4). To fulfil the requirements of a usable semantic search system that supports natural language queries and handles the limitations of existing systems, there is a need of a new framework that has the capability for

- SPARQL Query Translation
  As discussed earlier (in Section 4.4.1), a semantic search system needs to translate natural language queries to SPARQL queries to improve the accessibility of semantic datasets for the end users.

- Natural Language Processing
  To be able to translate natural language queries to SPARQL queries, a semantic search system requires processing of natural language.

- Portability
  Portability (as discussed in Section 4.4.2) is a required feature to enable a semantic search system to work with different datasets. Existing semantic search systems

(such as PowerAqua, FREyA and QAKiS) accommodate the feature of portability such that these can be installed and used with different individual datasets.

- Extensibility
  The vision of semantic search is to access data from multiple datasets across the web. Such a search system should be able to merge multiple ontologies (as discussed in Section 4.4.3) to fetch data from multiple domains for a specific user query.

- Result Optimisation
  Since the results retrieved from a SPARQL endpoint are mostly the URIs of resources, it causes poor readability and visualisation. Also, result ranking poses challenges while querying multiple datasets simultaneously. An effective semantic search tool should cater for better readability and result ranking.

- Interoperability
  To enable applications to utilise semantic datasets, there is a need for an interface that acts like an API (Application Program Interface) to promote interoperability (as discussed in Section 4.4.4).

4.6 Summary

Structured data from semantic knowledge bases can be accessed using either a complex formal query language like SPARQL or end user interfaces (e.g. form-based, menu-based, view-based, natural language based etc.). Usability study of user interfaces (for semantic search) has shown that the natural language interfaces are the most liked choice. Currently, there are a number of semantic search NLIs available but they have a number of limitations e.g. usability, portability, scalability etc.

An analysis of the existing search systems have led the author to conclude that there is a need for a new framework for semantic search. This chapter has also identified the features for the proposed system that will be used as an evaluation criteria.

The next chapter introduces a new framework proposed by the author and provides an in-depth explanation of individual modules of the author's framework.

Chapter 5

SIRF - A New Framework for Semantic Search

5.1 Introduction

Internet users are accustomed to having a good user experience with existing search engines (e.g. customisation, saving search history, auto-correct and auto-suggest options) even though that does not necessarily involve the best-matched results. Although the main focus of a semantic search is to retrieve best matched results with highest relevancy and accuracy it is imperative that the user-friendly features of conventional search tools are retained in any semantic search tool implementation.

Currently, search tools can be divided into two categories i.e. (i) syntax based and (ii) semantic based. For end users, it is not convenient to use two different tools for different purposes. Since there are already a number of syntax based or semi-semantic search tools that have good user friendly features, there is a need for a semantic search framework that can be implemented individually as well as integrated with other search tools. Such a system should be using a Natural Language Interface for domain independency. Natural Language based semantic search is a challenging problem in information retrieval. Existing solutions are limited in various aspects as discussed in Chapter 4. This chapter proposes a new framework called SIRF (**S**emantic **I**nformation **R**etrieval **F**ramework) that facilitates an effective search over the semantic datasets. Section 5.2 defines the key concepts of the proposed framework (SIRF) followed by Section 5.3 that provides the detailed architecture of the framework. Section 5.4 summarises the overall workflow of the framework. The chapter finishes with a summary in Section 5.5. Parts of this chapter have been taken from the author's published work [Fatima *et al.*, 2014a] and [Fatima *et al.*, 2015a].

5.2 Conceptual Architecture of SIRF

From a comprehensive study of existing semantic search systems (as described in Chapter 4), the author concluded that the main reason of the inadequacy of these tools is lack of a proper framework. The author worked through the problems with the existing systems and proposed the new framework SIRF. Since SIRF deals with semantic search, it needs to manipulate RDF, ontologies and query languages for RDF. To meet the above requirement, the author introduced certain key concepts (as defined below) to constitute the conceptual architecture (Figure 5.1) of SIRF.

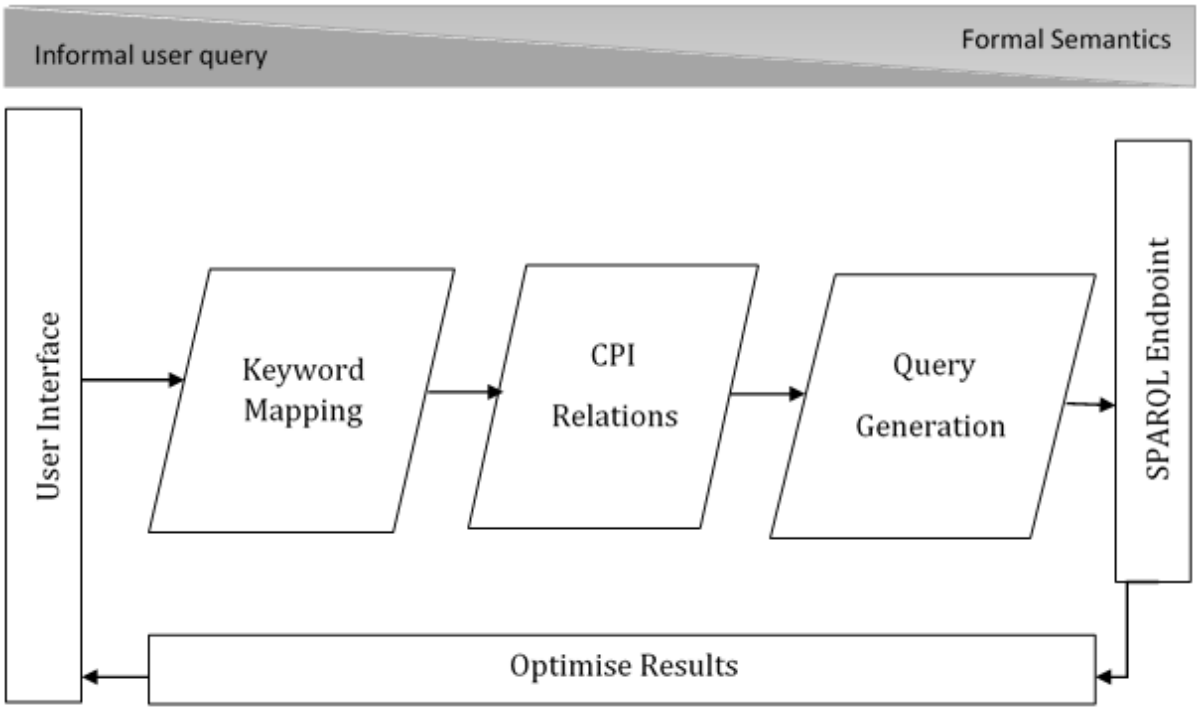


Figure 5.1: Conceptual Model of SIRF (Source: Author, 2015)

5.2.1 CPI Tags

The term CPI has been derived from three main concepts in Ontology i.e. Class, Property and Instance (CPI). SIRF uses a CPI tag as an identifier to tag keywords. Each keyword may belong to a CPI triple (C, P, I) where 'C' denotes a class, 'P' denotes a property and 'I' denotes an Instance. The stored CPI tags can be used for keyword tagging, concept mapping and, auto-suggest. For instance, for a query "Da Vinci", the system can produce two CPI tags (i) (Movie, name, Da Vinci) and (ii) (Book, title, Da Vinci). The generated CPI tags provide important information about the term "Da Vinci" e.g. "Da Vinci" is an instance that belongs to two classes *Movie* and *Book*.

5.2.2 Caching

For a graph as large as the Internet millions of iterations may be necessary to search for a matching result and this raises the problems of query termination, processing time and cost of processing. To deal with these issues search engines normally use a back-end activity where searchbots and crawlers navigate the Internet and retain an updated cached copy of the web pages. To enhance the efficiency of information retrieval, the conventional search engines deploy indexing [Manshadi and Li, 2009], caching and recall techniques. The quality of query answers depends on the source selection (i.e. selecting potentially relevant data) [Naseer *et al.*, 2013] but the quality of search tool depends upon the speed of informal retrieval. Using the above techniques speeds up the process though at the cost of more storage space. SIRF proposes two internal caches to speed up the process of information retrieval as defined below.

5.2.2.1 Bag of Keywords

The researcher introduces the term 'Bag of Keywords' to denote a repository to save keywords retrieved from other semantic knowledge bases. The 'Bag of Keywords' caches these keywords with related CPI tags. Caching keywords with CPI tags helps in multiple ways. For example:

- Named Entity Recognition (NER)
  [Mikheev *et al.*, 1999, p. 1] define Named Entity (NE) recognition as a process that "involves processing a text and identifying certain occurrences of words or expressions as belonging to particular categories" e.g. Persons, Organisations, Books etc. NER is a very important technique of natural language processing. For instance, for a query "who is the writer of Pride and Prejudice?", if *Pride and Prejudice* is not recognised as a named entity, any language processor will assume it is three different keywords hence producing wrong results. SIRF caches NEs in the "Bag of Keywords" to map with a user query.

- Efficiency
  To recognise NEs and map keywords with CPI tags using cached data is much faster and reliable than using external APIs.

- Concept Mapping
  Caching terms in the "Bag of Keywords" also helps to map similar concepts e.g. a CPI tag (Person, name, arooj) suggests that "arooj is a person's name" and it can be matched to different domains having *Persons* data.

5.2.2.2 Ontology Cache

The Ontology Cache is a repository of aligned ontologies that is used to map query terms to their related ontology concepts. The process of ontology caching will be explained in detail in Chapter 6.

5.3 Architecture of SIRF

SIRF final architecture evolved from multiple design and test phases. The author started with a big picture and proposed a framework that handles semantic search for RDF and non-RDF data (such that the non-RDF data is tagged with related ontology concepts). The author identified the required components for an effective framework (that handles semantic search) i.e. user interface, RDF processor, ontology processor, cache and a query processor.

The first version of the framework ([Fatima *et al.*, 2014a]) consisted of User Interface, Ontology Processor, Query Optimiser, Cached Repository and Document Identifier. The Document Identifier cached RDF and non-RDF files in the Cached Repository. The User Interface accepted keywords only queries. The user query was checked for ambiguities by the Query Optimiser and results were fetched from the Cached Repository against the keywords. The drawbacks with this version were (i) the space required to cache a repository as big as world wide web (ii) the keywords based queries and (iii) relying on local cached pages for results.

The author revised the framework in a second phase ([Fatima *et al.*, 2014b]) to replace the Cached Repository with the Ontology Cache and the Bag of Keywords. The proposed repositories cached tags for ontology concepts and instances, hence saving the storage space and ensuring more up to date results. Since it was not the intention of the author to handle non-RDF tagging, the research scope was narrowed down to deal with RDF data only (using SPARQL endpoints). In the third phase ([Fatima *et al.*, 2015b]), the author added the Result Optimiser to the framework to display the results ordered according to the ontology ranking. In the fourth phase, the author started working on free-text user queries to support keyword-based and natural language queries and expanded the Query Optimiser to add the Syntax Analyser and the Semantic Analyser.

This section defines the latest version of author's proposed framework (SIRF) (as shown in Figure 5.2) and explains its individual modules.

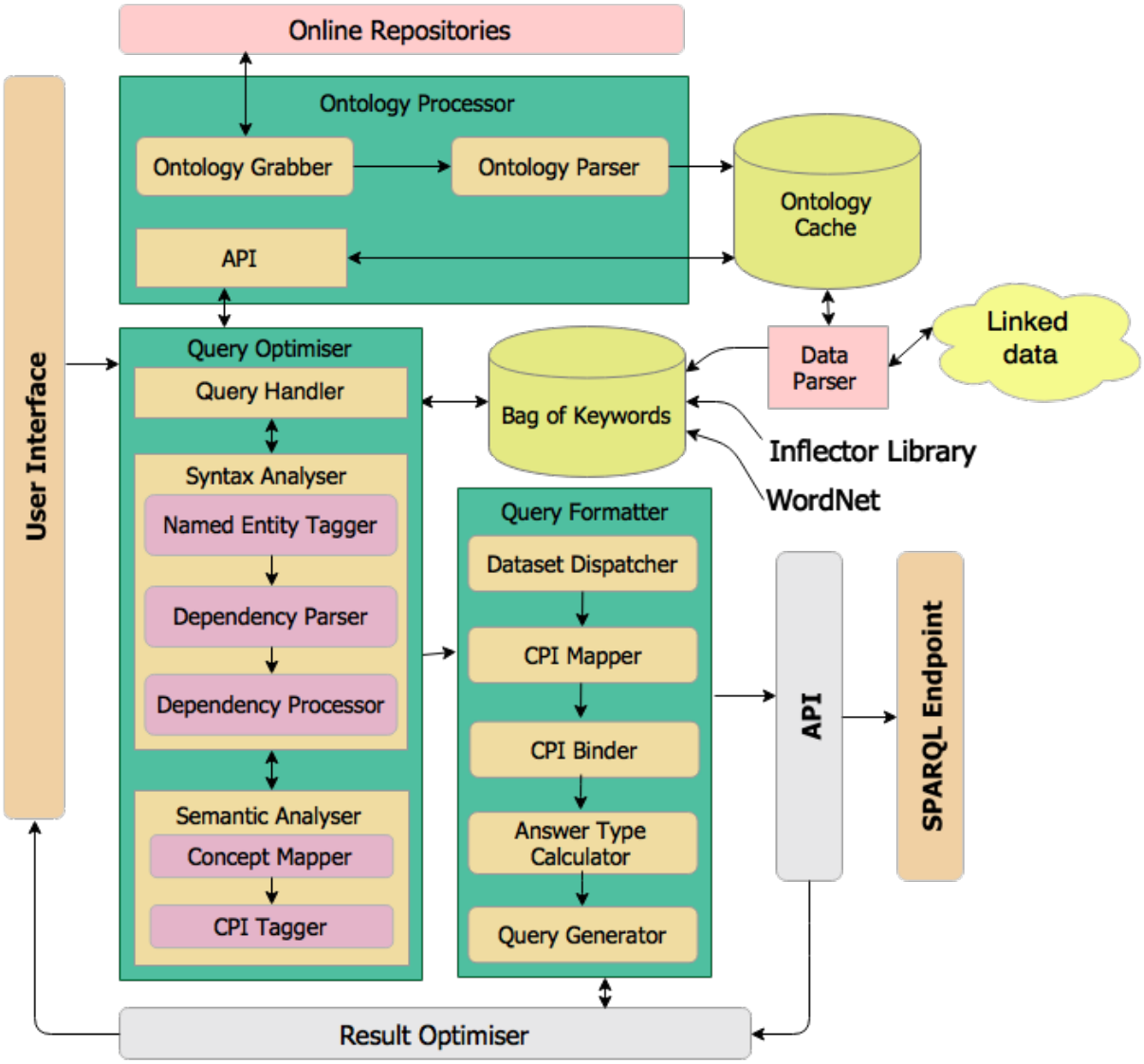


Figure 5.2: Detailed Architecture of SIRF (Source: Author, 2015)

5.3.1 Data Parser

Considering that there is a huge amount of data across the Internet that is not in RDF format, alternative strategies are needed to search such data resources. It is not easy to migrate all non-RDF data over the Internet into RDF, and convincing the data owners to migrate their data into RDF format is also a big challenge (owners may not understand the benefits of the Semantic Web and the benefits may not be evident until the Semantic Web is realised - a circular problem). Therefore to make the Semantic Web fully functional in the short term, both RDF and non-RDF data must be accessible until such a time that most of the Internet data is converted to RDF, or until some appropriate conversion techniques are found. There are a number of research efforts already proposed in this area

e.g. [Gerber *et al.*, 2013] and some APIs such as RDFTranslator[34]. RDF can be embedded into languages like PHP[35] and HTML[36]. Also there are tools that can tag concepts in non-RDF documents e.g. SenticNet[37]. SIRF provides the Data Parser module that currently supports only RDF format and may be extended in future to handle non-RDF data. This module is a very important part of SIRF as it sets up the core Knowledge Base of the framework.

5.3.2 User Interface

Usability is one of the main challenges for a semantic search tool. A query engine can only work effectively if it actually knows what to search for. It is generally hard to guess what the user wants to search for if the user query is not specific. For example if a user types in the word 'Bat', it is difficult to understand if user is searching for the meanings of the word or the animal named as 'Bat' or the cricket bat or everything about the word 'Bat'. Different search engines behave differently for a search query and deal with it using their particular algorithms. A number of existing search engines keep their users happy by caching their previous searches, ranking the searches in geographical order and learning user preferences. Yet, it is a challenge to understand the user query when it comes to a specific request.

In a conventional programming environment (such as relational databases), the developers will write queries knowing the search criteria, data structure and the query language. However a truly optimised semantic search requires a search engine that can itself write queries to be implemented by the query engine. However a query language alone is not sufficient for a usable search system, it also depends upon the base data structure that it works upon and the top-level system interacting with it. This functionality must not compromise the qualities of a conventional end-user interface and should include user-friendly features such as auto-suggest and auto-correct functionality [Selvan *et al.*, 2012] and an ontological classification of words possibly by query-tagging [Li and Wang, 2010].

[34] http://rdf-translator.appspot.com/ - a tool to convert one RDF format to another RDF format
[35] Hypertext Preprocessor
[36] HyperText Markup Language
[37] http://sentic.net/ - a tool to extract ontology concepts from plain text.

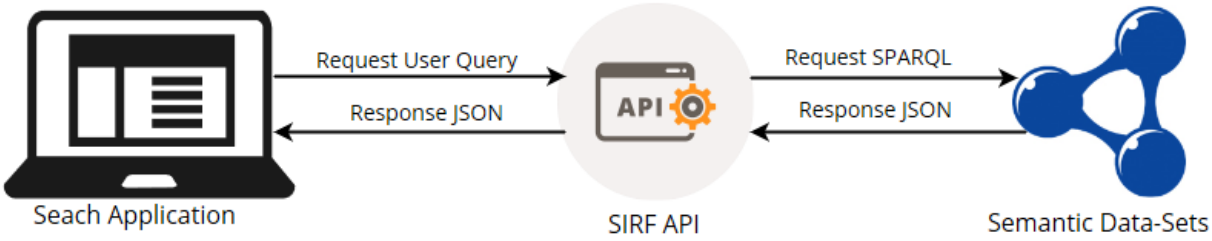


Figure 5.3: SIRF as an API (Source: Author, 2015)

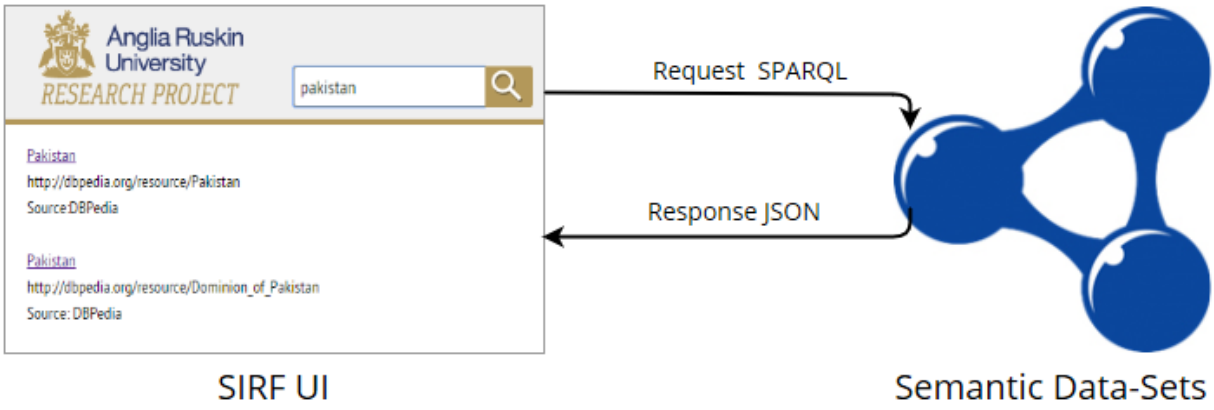


Figure 5.4: SIRF UI View (Source: Author, 2015)

A user interface in this context, is an entry point that accepts a natural language query and returns matching results. The user interface for SIRF plays two roles here. It could be used as a direct entry point for user queries (Figure 5.4) or it can act as an API (Application Programme Interface) (Figure 5.3). An API is a set of tools for building software applications such that, one application can be consistent with another application [Minnaert *et al.*, 2002]. Having set up as an API, SIRF can be integrated with other search tools. When using SIRF as a user interface, the core system will be able to handle semantic search queries only. A system that is already handling syntax based queries and able to categorise the syntax based and semantic queries, can provide a fuller search experience by integrating SIRF as a semantic tool.

In the present work the researcher proposes an interface that provides a facility to tag search terms (keywords) with their related ontology concepts. At a basic search level, the interface smartly re-phrases the keywords based on its own understanding and computes possible interpretations of a user query. In the case multiple interpretations are computed, the User Interface displays or returns (to software agent) results matched to the most used interpretation and provides the remaining interpretations as a suggestion to the end-user or the software agent. For example for a search on java and apple, *java* can be interpreted as {language or place} and *apple* can be interpreted as {fruit or company}.

5.3.3 Query Optimiser

A major challenge for a semantic search engine is to translate a natural language query to a search query that is understandable by the query engine (i.e. SPARQL). The processing of the semantic query language is different from other query languages because it does not work on pre-written queries by program developers that simply match the search criteria with different pre-defined data fields (as in relational database) or nodes (like XML). Instead it dynamically writes queries using various algorithms at initial query runtime.

To write runtime queries, the user keywords should be mapped to their relative ontologies. The same approach has been explored by [Malik and Rizvi, 2012] where they modelled some test inputs entered by the user with a related ontology, although it is not clear from their paper how the input is supposed to be entered. End-users are rarely specific when writing search keywords and always expect the search engine to understand what they are looking for.

The query optimiser consists of three main parts as follows:

5.3.3.1 Query Handler

This is the first part of the module to interact with the user query. It checks if a SPARQL query has already been cached for the same or similar query. If the exact user query has been cached, it simply sends the results back to the user interface. If a similar query is cached, it confirms from the user if that was the intended query. Similarity of the query is calculated by matching the concepts in both queries (user query and cached query). In the event of the query handler not finding any cached query, it sends the user query to the Syntax Analyser.

5.3.3.2 Syntax Analyser

Extracting semantic relations among terms in a search query is not always an easy task. For a simple query that only asks for a single entity (e.g. Person, Books, Cities etc.) or single entity and single attribute (e.g. Simon's books), it is relatively straightforward for the tool to attach ontology tags and find concept relations. The task of finding relations among concepts becomes difficult for a search tool (that supports natural language queries) while dealing with complex queries. For a query like "What is the first name of the author and surname of the editor of Ride a Rhino?", it can be difficult for the semantic tool to attach correct properties to the related entities. For the given example (above), a semantic analyser may be able to identify the concepts like "first name",

"surname", "author", "editor" and "Ride a Rhino" but it may not link them together based on given syntax i.e. the property "first name" should be attached to "author" and "surname" should be attached to "editor". To achieve this purpose, SIRF needs natural language processing practices. Since the SIRF can be integrated with other search systems, it can delegate natural language processing to be handled by the top level system but this approach may lead to extra integration steps. Hence the query optimiser contains a "Syntax Analyser" to perform basic syntax analyses to find term dependencies. The Syntax Analyser has three sub modules (i.e. the Named Entity Tagger, the Dependency Parser and the Dependency Processor) that will be discussed in Chapter 7.

5.3.3.3 Semantic Analyser

The Semantic Analyser scans the keywords entered by the end user and maps them to the related terms available in the "Bag of Keywords". This module also identifies the ontology concepts defined in a user query i.e. classes and properties. The Semantic Analyser has two sub modules (i.e. the Concept Mapper and CPI Tagger) that will be further discussed in Chapter 7.

5.3.4 Ontology Processor

The Ontology Processor plays a vital role for query optimisation. It will help the Query Optimiser to suggest tags to query keywords where tags have associated ontologies.

The retrieving of ontologies from an online store or library (every time a user enters search input) in order to tag a word is a lengthy process that will have a cost in terms of efficiency. Unavailability or slow response of an online ontology server can also limit the functionality of the search engine. To overcome this issue, the proposed Ontology Processor consists of three parts:

1. **Ontology Grabber**
   One part of the Ontology Processor is called the Ontology Grabber that serves like a 'cron job' (that is, a scheduled job on a server), which regularly updates an ontology list from an ontology store.

2. **Ontology Parser**
   The second part of the Ontology Processor parses the ontologies retrieved by the Ontology Grabber and retains a cached copy of the parsed ontology concepts. The grabbed concepts are placed in an Ontology Cache.

3. **API Service**
   The third part of the Ontology Processor works like an API (Application Program Interface) that receives key words from the Query Optimiser and returns matching tags with associated ontologies as a response.

The algorithm for ontology processing will be explained in Chapter 6.

5.3.5 Query Formatter

The Query Formatter is the core part of SIRF. Of the different semantic query engines that have been proposed, SPARQL is the one existing standard that supports the W3C standard and is continuously being updated by a SPARQL working group[38]. For this reason, the Query Formatter supports SPARQL language only and translates user queries to SPARQL queries. The Query Formatter consists of the following parts:

1. **Dataset Dispatcher**
   This section of the module is responsible for generating a list of datasets that have matching concepts to a user query.

2. **CPI Mapper**
   This section of the module maps tagged CPI values from the Query Optimiser to match similar concepts across heterogeneous datasets.

3. **CPI Binder**
   This section finds the declared relations and the missing relations among given concepts. Having all the concepts and relations in-line, it generates simple statements to declare relations.

4. **Answer Type Calculator**
   This section of the module attempts to find the intent of a user query.

5. **Query Generator**
   This section translates the statements from the above steps to SPARQL syntax and generates SPARQL queries for the related datasets.

The sub modules of the Query Formatter (as defined above) will be detailed in Chapter 8.

[38]http://www.w3.org/2011/05/sparql-charter - SPARQL Working Group Charter

5.3.6 Result Optimiser

For this work, the author considers two main aspects of result optimisation i.e. (i) result ranking and (ii) result readability. The Result Optimiser optimises the results using pre-defined schema that increase the readability and visualisation of search results.

5.3.6.1 Result Ranking

Semantic data ranking is a challenging task as a semantic search tool sees the Internet as a giant graph. To rank results there are various techniques that can be used i.e. frequency-based weights [Du and Hai, 2013] or record of trusted web pages [Bard and Ferrara, 2011]. Commercial search engines like Google build trust level based on different techniques such as hit counts, bounce rate and listing authenticated domains etc. In addition to these techniques, a result ranker may use other approaches using various customised criteria (e.g. order of relevance, order of occurrence, order of published date, user location) as used by Google Scholar [Beel and Gipp, 2009].

SIRF adopts ontology based concept ranking. For a given concept, the Result Optimiser first lists all namespaces (which define that concept) in order of their usage frequency. Second, it ranks domains for each namespace based on the number of instances that domain maintains for the given concept. The process of result ranking will be detailed in Chapter 9.

5.3.6.2 Result Readability

The SPARQL endpoint returns results in JSON[39], XML[40] and HTML[41] format and mostly it contains URIs for the resources. Since the proposed system is intended to be end-user focused the results need to be readable and visually user friendly. To achieve this the author introduces a result schema. The schema defines key attributes for each class in an ontology with a pre-defined layout (as shown in Table 5.1).

The query processor uses schema defined for a class while generating a SPARQL query e.g. for the class 'Country' the schema attributes defined is country name and the user interface displays the result based on schema defined. Figure 5.5 and Figure 5.6 display the difference of a raw SPARQL result and a schema-based result. The detailed algorithm for using result schema will be explained in Chapter 9.

[39] JavaScript Object Notation
[40] Extensible Markup Language
[41] Hypertext Markup Language

Class	Attributes	Schema layout
http://xmlns.com/foaf/0.1/Person	name, givenName, familyName	name(familyName, givenName)
http://purl.org/ontology/bibo/Book	title	title
http://dbpedia.org/ontology/Country ...	name ...	name ...

Table 5.1: Result Schema (Source: [Fatima *et al.*, 2015b]

country	cname
http://dbpedia.org/resource/Pakistan	"Pakistan"@en
http://dbpedia.org/resource/Dominion_of_Pakistan	"Pakistan"@en

Figure 5.5: Example of raw SPARQL Result (Source: Author, 2015)

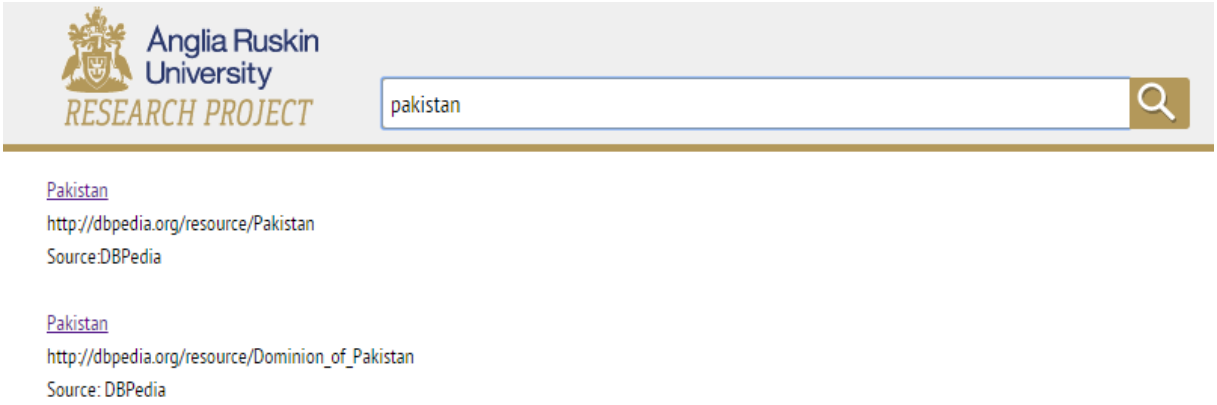


Figure 5.6: Example of Schema-based Result (Source: Author, 2015)

5.4 Work-flow of SIRF at a glance

SIRF has two main levels of processing. At the first level, the framework builds the core Knowledge Base of the system. The creation of this Knowledge Base will be discussed in Chapter 6 in detail. At the second level, the framework deals with user queries. Figure 5.7 summarises the work flow of SIRF to process a user query and generate optimised results.

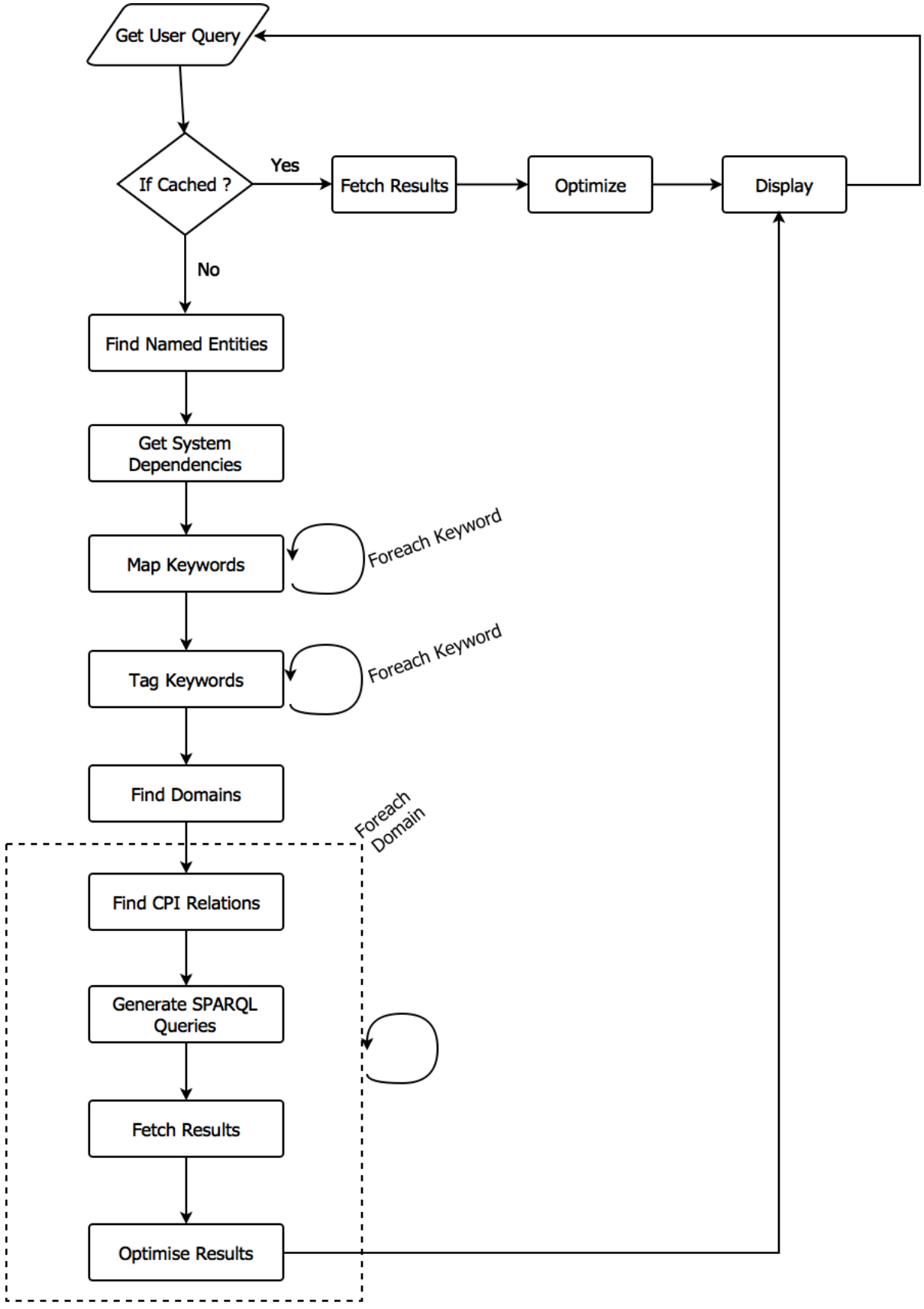


Figure 5.7: Flow Diagram for Processing a User Query (Source: Author, 2015)

5.5 Summary

This chapter introduces a new framework (i.e. SIRF) for semantic search. The framework has a number of modules (i.e. Data Parser, Ontology Processor, User Interface, Query Optimiser, Query Formatter and Result Optimiser) that work collectively to translate a natural language user query to SPARQL query.

Chapters 6 to 9 expand on the significant modules of SIRF. The next chapter explains the parts of SIRF that constitute the Knowledge Base of the framework.

Chapter 6

Knowledge Base of SIRF

6.1 Introduction

This chapter expands on parts of SIRF (as described in Chapter 5) that constitute the core knowledge base of the system. Parts of this chapter have been taken from the author's published work [Fatima *et al.*, 2014b]. SIRF manages its knowledge base by caching ontologies and maintaining a "Bag of Keywords". Section 6.2 explains the requirement for a Knowledge Base (KB) and highlights the concepts that build the Knowledge Base of the framework. Section 6.3 describes the process of data parsing followed by Section 6.4 that explains the process of ontology parsing.

6.2 Knowledge Base Management

For a semantic search system, it is necessary to understand the core structure of the system to be queried. In response to a user query, such a system should be able to determine answers to the following questions

- What information does the user intend to retrieve?
- Which systems have the intended information?
- What namespaces do they use?
- What are the web addresses for their SPARQL endpoints?

To answer the above questions, a semantic search system should have a knowledge base containing ontologies (for keyword mapping to find the intent of a user query), a record of

datasets and a record of namespaces used by individual datasets. Since SIRF is intended to be portable and integrable with other search systems, it introduces a flexible knowledge base. The framework caches a list of RDF datasets (currently entered manually) with the URL of their SPARQL endpoints and the supported version of SPARQL (as shown in Figure 6.1). The framework can be initialised with a list of any number of datasets and this feature makes it flexible to be used with closed domain[42] as well as open domain[43] systems. The knowledge base of SIRF is built in two steps i.e. (i) Data Parsing and (ii) Ontology Parsing.

id	domain	endpoint	version
1	http://www.bl.uk/	http://bnb.data.bl.uk/sparql	1.0
2	http://dbpedia.org/	http:/dbpedia.org/sparql	1.1

Figure 6.1: List of RDF datasets (Source: Author, 2015)

6.3 Data Parsing

This process is handled by the Data Parser module (as defined in Chapter 5) that uses an RDF parser to read RDF files from the listed domains. As the first step, the RDF Parser extracts the list of all namespaces from an RDF file and places a copy in the Ontology Cache to be parsed by the Ontology Processor. Figure 6.2 shows the list of namespaces in an example RDF file (highlighted by the square bracket).

[42]Closed domain systems are restricted to a specific domain e.g. music, education
[43]Open domain systems deal with multiple domains or all domains using a universal ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dct="http://purl.org/dc/terms/"
xmlns:isbd="http://iflastandards.info/ns/isbd/elements/"
xmlns:skos="http://www.w3.org/2004/02/skos/core#"
xmlns:bibo="http://purl.org/ontology/bibo/"
xmlns:rda="http://rdvocab.info/ElementsGr2/"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:blt="http://www.bl.uk/schemas/bibliographic/blterms#"
xmlns:bio="http://purl.org/vocab/bio/0.1/"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:event="http://purl.org/NET/c4dm/event.owl#"
xmlns:org="http://www.w3.org/ns/org#"
xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:interval="http://reference.data.gov.uk/def/intervals/"
xmlns:time="http://www.w3.org/2006/time#"
>

<rdf:Description rdf:about="http://bnb.data.bl.uk/id/resource/000044704">
<rdfs:label>Ezra Pound and T. S. Eliot / Richard Aldington</rdfs:label>
<blt:bnb>GB5415875</blt:bnb>
<owl:sameAs rdf:resource="http://bnb.data.bl.uk/id/resource/GB5415875" />
```

Figure 6.2: Namespaces from an RDF file (Source: Author, 2015)

In the second step, the RDF parser fetches all instances and Named Entities from the dataset and places them in the "Bag of Keywords" (Figure 6.3).

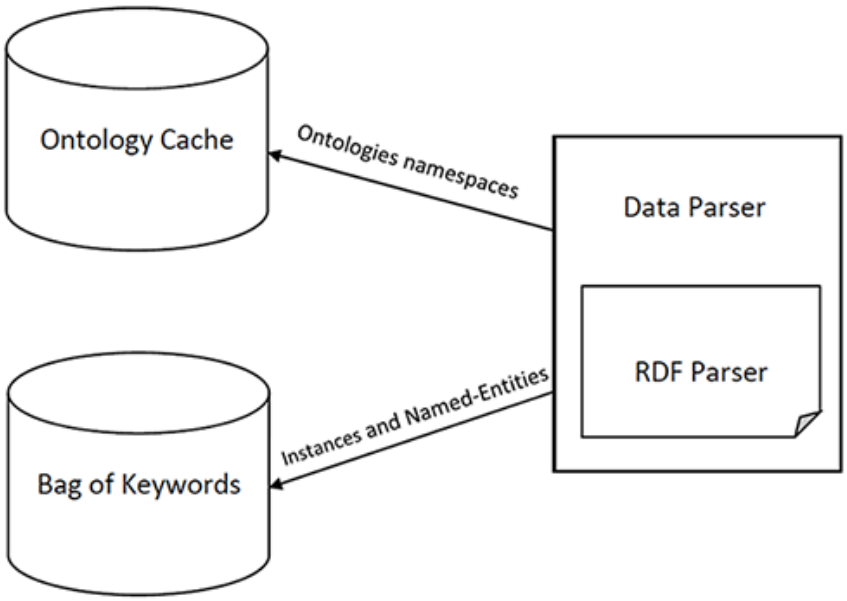


Figure 6.3: Data Parsing (Source: Author, 2015)

6.4 Ontology Parsing

As discussed in Chapter 5 and particularly in Section 5.3.4, the Ontology Processor in SIRF plays a vital role for efficient information retrieval. The Ontology Grabber (Figure 6.4) gets the URLs of cached namespaces and pulls them from online repositories (using Algorithm 1).

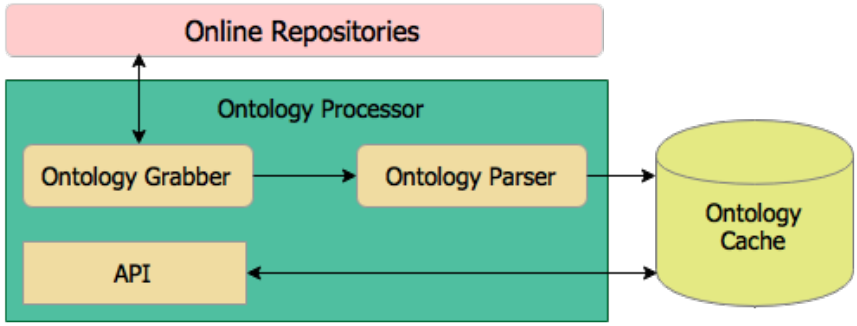


Figure 6.4: Ontology Processor (Source: Author, 2015)

Algorithm 1 Ontology Caching

```
Input: ontology_url, cache_path
records ← count(ontology_list)
lists ← new queue
while record+1 < records do
  file_exists ← check_if cached(listrecord,cache_path)
  if file_exists then
    creation_date ← get_creation_date(listrecord)
    modify_date ← check_modify_date(listrecord)
    if modify_date > creation_date then
      file ← fetch_revision(listrecord)
      update_index(file)
      update_cache(file)
    else
      //do nothing
    end if
  else
    file ← fetch_file(listrecord)
    insert_index(file)
    save_cache(file)
  end if
end while
```

For each given namespace, the Ontology Grabber checks if that namespace has already been cached. In the case of a cached record being found, it checks for any modifications made to the file after it was last accessed. In the event that the namespace was not parsed before or it has been modified, the Ontology Cache will be updated with new records. The new and updated records will be sent to the Ontology Processor for further processing.

The Ontology Parser parses a namespace and extracts all concepts used in it i.e. classes and properties. The parsed classes and properties are cached in the Ontology Cache for quick retrieval. Similar concepts are aligned in groups for ontology matching[44]. Different namespaces can use different names for the same concept or re-define the same concept e.g. the concept "Person" has been defined by a number of namespaces i.e. FOAF, DBPedia and Schema.org. This possibility raises the problem of ontology alignment. The ideal solution to align ontologies onto concept mapping, is to do it manually but it is not a practical solution especially with the rapidly increasing number of ontologies. SIRF

[44]Ontology Matching is a process to find correspondence or similarity between concepts.

uses OWL *sameAs* property to map similar concepts. SIRF adopts a learning behaviour for the concept alignment. When parsing a new namespace, it handles concept mapping in two ways. First, it attempts to place similar concepts in group. Second, if it finds a concept that uses *sameAs* for a grouped concept and an ungrouped concept, it merges all concepts in one group. This learning behaviour enables the framework to scale itself to handle heterogeneous datasets. The implementation of Algorithm 1 can be found in Appendix I (Listing 8). Algorithm 2 describes a basic procedure proposed by the author to group similar concepts. The implementation of Algorithm 2 can be found in Appendix I (Listing 3).

Algorithm 2 Concept Grouping

```
Concept_A ← main concept
Concept_B ← sameAs concept
if either of the concepts exist then
    group ← getGroup()
    assignGroup(Concept_A, Concept_B, group)
else
    group ← createGroup()
    assignGroup(Concept_A, Concept_B, group)
end if
```

In short, the Ontology Cache saves records for the followings

- List of domains (RDF datasets) (Figure 6.1)
- List of ontology namespaces with their corresponding domains (Figure 6.5)
- Parsed ontology classes and properties

An end user can enter various terms for the same property or class e.g. plural form or synonyms. A good example for this is the property 'dct:creator' that defines a book author. For an end user the word 'creator' may not be a choice but he may use various other search terms (e.g. author, authors, writer, writers or written by etc.) to find book creators instead of using term 'creator'. For this reason the proposed model contains mapped terms for properties and classes saved in the Ontology Cache (see Figure 6.6).

id	domain_id	namespace	short
11	1	http://iflastandards.info/ns/isbd/elements/	isbd
7	1	http://purl.org/dc/terms/	dct
8	1	http://purl.org/NET/c4dm/event.owl#	event
4	1	http://purl.org/ontology/bibo/	bibo
5	1	http://purl.org/vocab/bio/0.1/	bio
18	1	http://rdfs.org/ns/void#	void
14	1	http://rdvocab.info/ElementsGr2/	rda
10	1	http://reference.data.gov.uk/def/intervals/	interval
6	1	http://www.bl.uk/schemas/bibliographic/blterms#	blt
15	1	http://www.w3.org/2000/01/rdf-schema#	rdfs
13	1	http://www.w3.org/2002/07/owl#	owl
9	1	http://www.w3.org/2003/01/geo/wgs84_pos#	geo
16	1	http://www.w3.org/2004/02/skos/core#	skos
17	1	http://www.w3.org/2006/time#	time

Figure 6.5: Ontology Namespaces (Source: Author, 2015)

id	property	property_short	term
1	http://purl.org/dc/terms/creator	dct:creator	author
2	http://purl.org/dc/terms/creator	dct:creator	authors
3	http://purl.org/dc/terms/creator	dct:creator	writer
4	http://purl.org/dc/terms/creator	dct:creator	writers

Figure 6.6: Concept Mapping (Source: Author, 2015)

6.5 Summary

A semantic search tool needs to process ontologies and datasets to perform semantic queries. To process ontologies on the fly can be a tedious task and it can raise a number of issues i.e. time-out, in-accessibility of resources and slow response. To solve these issues, the author proposes to have a Knowledge Base to cache core concepts. The Ontology Processor saves classes and properties in the Ontology Cache and collects aligned concepts in groups.

The Bag of Keywords maintains a repository of instances and Named Entities. This chapter has explained the processes of data parsing and ontology caching to build the Knowledge Base of SIRF (i.e. Ontology Cache and Bag of Keywords).

Having explained the proposed framework and its Knowledge Base, the next chapters

7 to 9 will focus on the process of translating a user query to a SPARQL query. The first step to achieve user query translation, is to process natural language text. The next chapter explains the steps involved in text processing for natural language user queries.

Chapter 7

Text Processing

7.1 Introduction

SIRF allows users to enter natural language queries bridging the gap between semantic web and the end users. To provide a natural language interface for semantic search is a very challenging task due to linguistic ambiguity (e.g. sentence structure, expression variations, synonymous etc.) and SPARQL compliance rewriting [Sharef and Noah, 2013]. This chapter describes pre-processing of natural language user queries expanding on the SIRF's modules that take part in the text processing. Section 7.2 describes various techniques for natural language processing and highlights the framework's modules involved in syntactic and semantic analysis. Section 7.3 gives a detailed overview of the process of syntactic analysis. Section 7.4 explains the process of semantic analysis followed by a summary of the chapter in section 7.5.

7.2 NLP

A natural language is a language spoken by people e.g. English, Swedish etc. In computer science, natural language processing (NLP) is a sub field of artificial intelligence that aims to make computers understand the text written in human language [Chopra *et al.*, 2013]. Almost all languages have their own syntax and grammar. There are a number of automated tools that can convert one natural language to another. Similarly, computer languages have their own syntax and grammar. To convert a natural language to a computer language, a tool needs to know the syntax of both languages and should be able to process NL. The most used types of NLP techniques are:

- **Morphological Analysis**
  In this type of analysis individual words are analysed into their components and non-word tokens (e.g. punctuation) are separated from the words. An example of morphological analysis is the word *running* that will be split into *run* + *ing*. Syntactic analysis uses morphological analysis to find sentence structure.

- **POS (Part Of Speech) Tagging**
  POS tagging is a technique that reads text and tags the terms with corresponding parts of speech like noun, verb and adjective [Collobert *et al.*, 2011].

  For a sentence "That is how a POS tagger works", a POS tagger will generate a text as follows:

  That\DT is\VBZ how\WRB a\DT POS\NNP tagger\NN works\VBZ

  Where
  DT = determiner
  VBZ = Verb 3^{rd} person
  WRB = Wh-adverb
  NNP = Proper singular noun
  NN = Singular noun

- **Stemming**
  Stemming is a tool in NLP that reduces inflected words to their base form. Some examples of the output generated by a stemming tool are

  Running ⇒ run
  Is, am, are ⇒ be
  Cat, Cats, Cat's, Cats' ⇒ cat

- **Syntactic Parsing**
  "A natural language parser is computer software that automatically performs parsing and outputs the structural description of a given string in the context of a specific grammar" [Kakkonen, 2007, p. 1]. NLP parsers are used to identify sentence structure and get different forms of text depending upon parser types. Dependency parsers are increasingly being used to parse natural language particularly to generate tree structures.

  For a sentence "List all authors born in 1945", a dependency parser (such as Stanford NLP dependency parser [Chen and Manning, 2014]) will generate the following output:

root (ROOT{0}, List{1})=> root element
det (authors{3}, all{2})=> determiner
dobj (List{1}, authors{3})=> direct object
acl (authors{3}, born{4}) => clausal modifier
case (1945{6}, in{5})=> case marking (e.g. prepositions)
nmod:in (born{4}, 1945{6})=> nominal modifier

In the proposed system, the natural language is processed in two steps, syntactic analysis and semantic analysis, handled by the Syntax Analyser and the Semantic Analyser respectively.

7.3 Syntactic Analysis

This is an analysis of words (in a user query) transforming them into structures to show which words relate to each other. Before converting a user query to a semantic query, it is necessary to identify the syntactic structure of the input sentence. Modern search engines are rich with syntax parsing algorithms e.g. phrase parsers, keywords identification, entity recognition etc. These features can be useful for syntactic analysis of the text as a pre-processing step for the Semantic Analyser. The Syntax Analyser (Figure 7.1) takes charge of syntactic analysis and processes natural language queries in three steps:

- Named Entity Tagging
- Dependency Parsing
- Dependency Processing

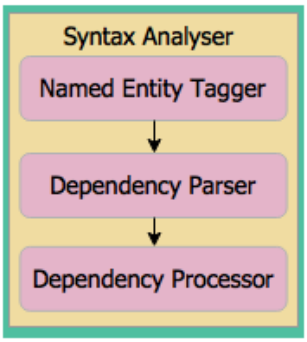


Figure 7.1: Syntax Analyser (Source: Author, 2015)

7.3.1 Named Entities Tagging

[Nadeau and Sekine, 2007, p. vi] define Named Entity Recognition as a field that "aims at extracting and classifying mentions of rigid designators, from text, such as proper names, biological species, and temporal expressions". The Syntax Analyser of SIRF tags Named Entities (NEs) from a user query as a first step of syntax analysis that helps the NLP parser to produce a proper parse. Without tagging NEs from the text, the parser may produce a wrong parse. For example Table 7.1 shows the output from a dependency parser for a sentence "who is the author of Ride a Rhino?" where the parser tagged "Ride a Rhino" as three different words (verb, determiner and noun). This kind of problem can be solved by "Named Entity Recognition (NER)" as an initial step of text processing.

Query	Who is the author of Ride a Rhino?
Output	root(ROOT{0}, who{1}) cop(who{1}, is{2}) det(author{4}, the{3}) nsubj(who{1}, author{4}) case(ride{6}, of{5}) nmod:of(author{4}, ride{6}) det(rhino{8}, a{7}) dep(author{4}, rhino{8})

Table 7.1: NLP Example Output (Source: Author, 2015)

There are a number of online APIs available for Named Entity Recognition e.g.

(i) Stanford Named Entity Tagger[45]

(ii) Alchemy Entity Extraction API[46]

(iii) Dandelion API[47]

(iv) Text Razor API[48]

All of the above APIs can only recognise three types of entities i.e.

[45]http://nlp.stanford.edu:8080/ner/process - Stanford Named Entity Tagger is an API tool to tag named entities in a text.

[46]http://www.alchemyapi.com/api/entity-extraction- Alchemy Entity Extraction API is a tool to extract entities from the given text.

[47]https://dandelion.eu/semantic-text/entity-extraction-demo - Dandelion API is a tool to extract entities from the text.

[48]https://www.textrazor.com/demo - Text Razor API is a tool to extract named entities from the text.

- Person
- Organisation
- Location

Naturally there are many more types of entities which are of interest to semantic search e.g. Books, Brands etc. Also, different APIs use different names for entities e.g. Organisation or Company, Location or Place. This ambiguity of entities and inability to recognise a wider set of entities limit their usability. The Data Parser module of SIRF extracts NEs while parsing the RDF datasets and caches in the "Bag of Keywords" (as described in Chapter 6). The Named Entity Tagger matches NEs from the text to the cached NEs and tags them accordingly. For example, for the above query (who is the author of Ride a Rhino?), the Named Entity Tagger tags "Ride a Rhino" with an identifier NE1_Book that helps the parser to read the text properly. Below is an example of parsed text after adding NE tags.

root(ROOT{0}, who{1})
cop(who{1}, is{2})
det(author{4}, the{3})
nsubj(who{1}, author{4})
case(NE1_Book{6}, of{5})
nmod:of(author{4}, NE1_Book{6})

7.3.2 Dependency Parsing

The two most used types of NLP parsers available are (i) dependency parsers [McClosky *et al.*, 2012] and (ii) constituency parsers [Vinyals *et al.*, 2014]. "Dependency parsers can recover much of the predicate-argument structure of a sentence, while being relatively efficient to train and extremely fast at parsing." [Ambati *et al.*, 2014, p. 159]. A constituency parser breaks a text into sub-phrases and identifies types of phrases. In contrast, a dependency parser connects words in a sentence to each other based on their relationship. The concept of a word-to-word link occurs naturally in consideration to support semantic relation between words [Covington, 2001]. The relevance of dependency parsing to semantic relations became the motivation to choose dependency parsing techniques to parse a natural language query. The manipulation of dependency parsing techniques to find dependent relations between text terms, distinguishes author's approach from existing semantic search tools.

7.3.3 Dependency Processing

The output generated by a dependency parser needs some further processing to generate simple statements to be consumed by the Semantic Analyser. Since the Semantic Analyser works on semantic concepts only, it is not able to identify syntactic relations. The output of the dependency processing is a set of unique statements. Each statement can have a maximum of two concepts (i.e. class, property or instance). For example for a query "List books published after 2011", the dependency processing will produce following output

pair(List{1}, books{2}),
pair(books{2}, published{3}),
pair(published{3}, 2011{5}) *case*=>after

The values in curly brackets {} are the position indexes for each word that act as a unique identifier for individual words e.g. index {2} will always refer to the keyword 'books'. To make things simpler, the same output can be read as given below.

(list, books),
(books, published),
after(published, 2011)

Section 7.3.3.1 shows the algorithm the researcher produced for generating dependent pairs to help the Semantic Analyser to avoid unnecessary processing. The implementation of Algorithm 3 can be found in Appendix II (Listing 16).

7.3.3.1 Algorithm for Syntactic Processing

Algorithm 3 Syntactic Processing: Parse Q ⇒ Raw text

```
Input: Q: Raw text
Output: S: set of processed phrases
stmts(type, values) ← parse(Q) : parsed nlp statements
while stmt ≠ ∅ do
  for each pair p in stmts do
    if p → type is "determiner" then
      ignorePair()
    else if p → type is "copula" then
      ignorePair()
    else if p → type is "adjectival modifier" then
      combinePair()
      shiftIndex()
    else if p → type is "compound" then
      processCompunds()
      shiftIndex()
    else if p → type is "case" then
      S ← joinStatementsWithCase()
    else if p → type is "reference" then
      replaceValue()
    else if p → type is "conjunction" then
      S ← combinePairs()
    else if p → type is "neg" then
      S ← applyNegation()
    else if p → type is "root" then
      S_category ← findRootCategory()
    else
      S ← getDefault()
    end if
  end for
end while

S ← removeRedundancy()
return S
```

7.3.3.2 Explanation of Algorithm 3

This section explains the different procedures (used in Algorithm 3) to process various types of statements generated by the NLP dependency parser. The definitions of NLP terms (in this section) have been derived from the Stanford Dependencies Manual[49].

(i) **Process Determiner and Copula**
Determiner defines a relation between the NP (Noun Phrase) and its determiner e.g. for a text 'the book', 'the' is the determiner of 'book'. Copula is a dependent of its complement e.g. 'is' makes a copula in a text 'woman is'. As the determiner and copula do not make a difference for semantic analysis, the algorithm ignore pairs of both types.

(ii) **Process Adjectival Modifiers**
Adjectival modifier of a noun phrase serves to modify the meanings e.g. red shirt, first name etc. The algorithm combines adjectival modifiers with their noun phrase to make it one word and shift index to the NP index. For a NLP pair amod (minister{5}, prime{4}) the algorithm will process it as follows
amod (minister{5}, prime{4}) ⇒ (prime minister{5})

(iii) **Process Compounds**
Compounds serve to modify the head noun. For example for a text "who is the author of Basic Physics Tutorials?" the NLP parser generates following compounds
compound (Tutorials{8}, Basic{6})
compound (Tutorials{8}, Physics{7})

The algorithm combines the sequential compounds and shift the index to the head NP and the output is "Basic Physics Tutorial {8}".

(iv) **Process Cases**
Case in NLP parsing serves to link phrases to each other using prepositions e.g. of, for, in, on, with etc. For example for a query "who is the author of Hamlet?", the NLP parser will generate following output

root (ROOT{0}, who{1})
cop (who{1}, is{2})
det (author{4}, the{3})

[49]http://nlp.stanford.edu/software/dependencies_manual.pdf - Stanford Dependencies Manual.

nsubj (who{1}, author{4})
case (Hamlet{6}, of{5})
nmod:of (author{4}, Hamlet{6})

The algorithm generates a statement using the head noun phrase, the preposition and the NP at -1 index to the preposition. For the above example the algorithm will generate a case statement as follows
Case(Hamlet{6}, of{5}) ⇒ author of Hamlet {4}

(v) **Process References**
A reference is a word that refers to a noun phrase e.g. which, that etc. For example, in a text "choroid, which is a part of the eye" which refers to choroid. The algorithm replaces the referred word with original noun phrase.

(vi) **Process Conjunctions**
Conjunctions are the terms that join phrases e.g. *and*, *or* etc. The algorithm generates links for conjunction to help the Semantic Analyser to generate conditions.

(vii) **Process Negation**
The algorithms tags a phrase for negation that helps the Semantic Analyser to generate negation conditions (detailed in Chapter 8 Section 8.8.1).

(viii) **Get Category from ROOT**
For well-structured questions, ROOT provides the category of the required answer. For example for a question "Who is the prime minister of USA?" the output from NLP parser will be as given as:

root (ROOT{0}, who{1})
cop (who{1}, is{2})
det (minister{5}, the{3})
amod (minister{5}, prime{4})
nsubj (who{1}, minister{5})
case (USA{7}, of{6})
nmod:of (minister{5}, USA{7})

7.3.3.3 Example

Table 7.2 describes a complete example of the steps taken by the Algorithm 3 to process parsed NLP statements for a query "Find all patients diagnosed with a disease in the

choroid, which is part of the eye".

	Current Statements	Processed Statements
Process Determiner and Copula	root (ROOT{0}, Find{1}) det (patients{3}, all{2}) dobj (Find{1}, patients{3}) acl (patients{3}, diagnosed{4}) case (disease{7}, with{5}) det (disease{7}, a{6}) nmod:with(diagnosed{4}, disease{7}) case (choroid{10}, in{8}) det (choroid{10}, the{9}) nmod:in(disease{7}, choroid{10}) nsubj (part{14}, choroid{10}) ref (choroid{10}, which{12}) cop (part{14}, is{13}) acl:relcl (choroid{10}, part{14}) case (eye{17}, of{15}) det (eye{17}, the{16}) nmod:of (part{14}, eye{17})	root (ROOT{0}, Find{1}) dobj (Find{1}, patients{3}) acl (patients{3}, diagnosed{4}) case (disease{7}, with{5}) nmod:with(diagnosed{4}, disease{7}) case (choroid{10}, in{8}) nmod:in (disease{7}, choroid{10}) nsubj (part{14}, choroid{10}) ref (choroid{10}, which{12}) acl:relcl (choroid{10}, part{14}) case (eye{17}, of{15}) nmod:of (part{14}, eye{17})
Get Root Category	root (ROOT{0}, Find{1}) dobj (Find{1}, patients{3}) acl (patients{3}, diagnosed{4}) case (disease{7}, with{5}) nmod:with (diagnosed{4}, disease{7}) case (choroid{10}, in{8}) nmod:in (disease{7}, choroid{10}) nsubj (part{14}, choroid{10}) ref (choroid{10}, which{12}) acl:relcl (choroid{10}, part{14}) case (eye{17}, of{15}) nmod:of (part{14}, eye{17})	root(Find{1}, patients{3}) acl (patients{3}, diagnosed{4}) case (disease{7}, with{5}) nmod:with (diagnosed{4}, disease{7}) case (choroid{10}, in{8}) nmod:in (disease{7}, choroid{10}) nsubj (part{14}, choroid{10}) ref (choroid{10}, which{12}) acl:relcl (choroid{10}, part{14}) case (eye{17}, of{15}) nmod:of (part{14}, eye{17})
Process Adverb Modifier	N/A	
Process Compounds	N/A	

Process Case Statements	root(Find{1}, patients{3}) acl (patients{3}, diagnosed{4}) case (disease{7}, with{5}) nmod:with (diagnosed{4}, disease{7}) case (choroid{10}, in{8}) nmod:in (disease{7}, choroid{10}) nsubj (part{14}, choroid{10}) ref (choroid{10}, which{12}) acl:relcl (choroid{10}, part{14}) case (eye{17}, of{15}) nmod:of (part{14}, eye{17})	root(Find{1}, patients{3}) acl (patients{3}, diagnosed{4}) case(diagnosed{4} with{5} disease{7}) case(disease{7} in{8} choroid{10}) nsubj (part{14}, choroid{10}) ref (choroid{10}, which{12}) acl:relcl (choroid{10}, part{14}) case(part{14} of{15} eye{17})
Process Reference	root(Find{1}, patients{3}) acl (patients{3}, diagnosed{4}) case(diagnosed{4} with{5} disease{7}) case(disease{7} in{8} choroid{10}) nsubj (part{14}, choroid{10}) ref (choroid{10}, which{12}) acl:relcl (choroid{10}, part{14}) case(part{14} of{15} eye{17})	root(Find{1}, patients{3}) acl (patients{3}, diagnosed{4}) case(diagnosed{4} with{5} disease{7}) case(disease{7} in{8} choroid{10}) nsubj (part{14}, choroid{10}) ref (choroid{10}, part{14}) case(part{14} of{15} eye{17})
Process Conjunction	N/A	
Remove Redundancy	root(Find{1}, patients{3}) acl (patients{3}, diagnosed{4}) case(diagnosed{4} with{5} disease{7}) case(disease{7} in{8} choroid{10}) nsubj (part{14}, choroid{10}) ref (choroid{10}, part{14}) case(part{14} of{15} eye{17})	root(Find{1}, patients{3}) acl (patients{3}, diagnosed{4}) case(diagnosed{4} with{5} disease{7}) case(disease{7} in{8} choroid{10}) ref (choroid{10}, part{14}) case(part{14} of{15} eye{17})
Final Output	root(Find{1}, patients{3}) acl (patients{3}, diagnosed{4}) case(diagnosed{4} with{5} disease{7}) case(disease{7} in{8} choroid{10}) ref (choroid{10}, part{14}) case(part{14} of{15} eye{17})	⇒ (Find, patients) ⇒ (patients, diagnosed) ⇒ with(diagnosed, disease) ⇒ in(disease, choroid) ⇒ (choroid, part) ⇒ of(part, eye)

Table 7.2: Prototype of Dependency Processing (Source: Author, 2015)

7.3.4 Test Examples

The author has run random user queries to test Algorithm 3. Table 7.3 shows the output of tests made to validate the process of syntax analysis.

User Query	**Dependency Processing**
List all authors born in 1945	(authors, born), (List, authors), in(born, 1945)
Find books published in New York	(books published), (Find, books), in(published, NE1_New_York)
search for a book with ISBN 9780729408745	(ISBN, 9780729408745), for(search, book), with(book, ISBN)
Which states border Texas	(border, states), (border, Texas)
Horror movies	(Horror, movies)
Find English songs	(find, songs), (English, songs)
population of cities in California	of(population, cities), in(cities, California)
which city is the largest city in Pakistan	(largest, city), in(city, Pakistan)

Table 7.3: Evaluation of the Syntax Analyser (Source: Author, 2015)

7.4 Semantic Analysis

Semantic analysis is the process of mapping words in a user query to the terms saved in the Knowledge Base. It should correctly find meanings of individual words through the way they relate to each other. Semantic analysis is not always possible without syntactic processing. For a simple query like “Neilsen’s articles” it is relatively straightforward to identify the entities and semantics of the terms (i.e. Neilsen as a Person and articles as an

Article) and eventually relate them. But it is not that direct for complex syntactic queries. To make the process of semantic analysis simpler, the Syntax Analyser generates simplified statements (as discussed above) where each statement can have a maximum of two concepts. Finding a relation between two concepts makes the process more transparent.

The process of semantic analysis is handled by the Semantic Analyser and CPI Binder (detailed in Chapter 8). The Semantic Analyser maps the query terms with their related ontology terms (i.e. classes, properties and instances) and CPI Binder links them with appropriate relations. A semantic analysis may produce multiple interpretations of a query e.g. Figure 7.2 depicts an example of semantic analysis for a query "Simon's books published in 1972" where a year can be mapped as a publication date, birth date or any other date.

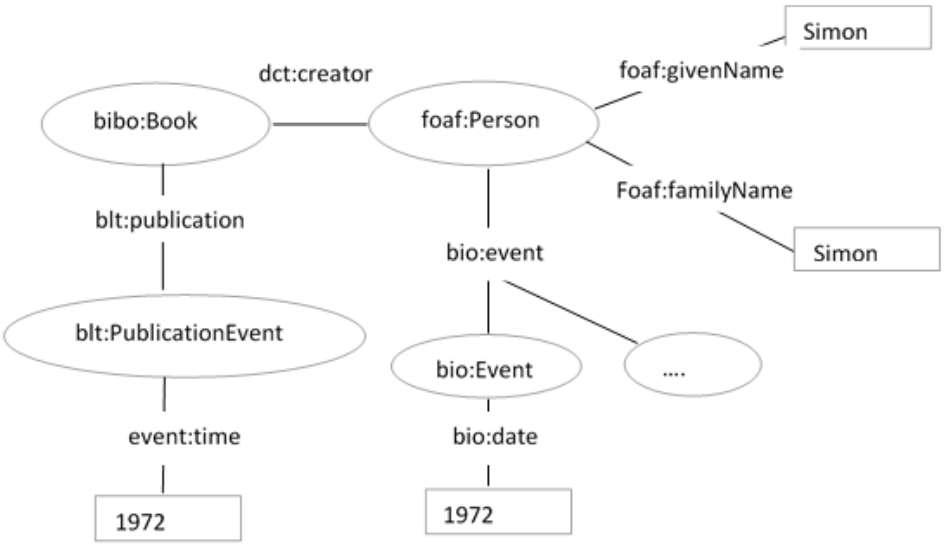


Figure 7.2: Example of keyword mapping and derived relations (Source: Author, 2015)

The process of syntactic analysis saves the CPI Binder from over processing multiple interpretations. For the example above (in Figure 7.2), the Syntax Analyser generates the following simplified statements

(Simon, books),
(books, published),
in(published, 1972)

The Semantic Analyser solves individual statements to map and tag keywords. The Semantic Analyser has two main parts i.e. (i) Concept Mapper and (ii) CPI Tagger (see details in the section below).

7.4.1 Concept Mapper

The Concept Mapper maps the terms in a user query to the terms mapped for ontology concepts in the Ontology Cache (as defined in Chapter 6 and specifically Section 6.4). For instance for a user query "List of book authors" the Concept Mapper will do the following mappings as shown in Table 7.4.

Keyword	Mapped Term	Mapped As
book	Book (bibo:Book)	Class
authors	creator (dct:creator)	Property

Table 7.4: Concept Mapping for Keywords (Source: Author, 2015)

7.4.2 CPI Tagger

The CPI tagger dispatches CPI tags (as defined in Chapter 5) to the user query terms to identify their semantics. It extracts classes, properties and instances from the search text e.g. for a query "simon's books" the keyword mapping will be
class = {Book},
properties= {null} and
instances = {(Person, name, simon)}.

7.5 Summary

This chapter has explained the process of text processing, paving the way to translate natural language user queries to SPARQL queries. SIRF performs NLP in two steps i.e. (i) syntactic analysis and (ii) semantic analysis. The Syntax Analyser parses NL query and generates simplified phrases. The Semantic Analyser maps the terms in a user query to their related semantic concepts. Figure 7.3 summarises the text processing performed in the framework.

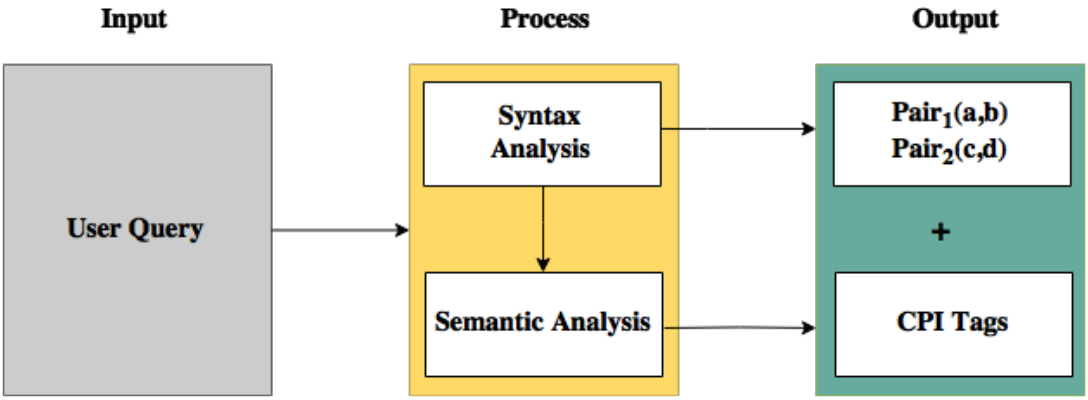


Figure 7.3: An overview of Text Processing (Source: Author, 2015)

After all the syntactic and semantic processing is done, the next step is to build a valid SPARQL query that will run across SPARQL endpoints and fetch data. The next chapter explains the process of generating SPARQL queries.

Chapter 8

Generating SPARQL Queries

8.1 Introduction

SPARQL[50] is a standard query language and protocol recommended by World Wide Web Consortium (W3C)[51] that can retrieve and manipulate data from RDF datasets [Harris *et al.*, 2013]. SPARQL 1.0 became a recommendation to query RDF datasets in January 2008. The author initially experimented with many queries in SPARQL 1.0 and later modified the framework to utilise features of the newer version i.e. SPARQL 1.1 that became a recommendation in March, 2013 [DuCharme, 2013]. Currently SIRF is capable of handling both versions of SPARQL which means it can query SPARQL endpoints of either version (though the SPARQL endpoints using version 1.0 can answer limited types of queries).

This chapter explains the process of translating parsed user queries (as explained in Chapter 7) to SPARQL queries. Section 8.2 describes the general syntax for a SPARQL SELECT query. Section 8.3 expands on SIRF's modules that undertake query translation. Section 8.4 defines the Dataset Dispatcher module to list datasets to be queried. Section 8.5 explains the process of concept mapping for listed datasets. Section 8.6 illustrates the process of CPI binding. Section 8.7 presents an algorithm to calculate answer type for the SPARQL query. Section 8.8 explains the process of generating SPARQL statements followed by a working example of query generation (Section 8.9). Finally, Section 8.10 explains the process of running SPARQL queries. The chapter finishes with a brief summary in Section 8.11.

[50] http://www.w3.org/TR/sparql11-overview/ - SPARQL Protocol and RDF Query Language (SPARQL).

[51] http://www.w3.org/ - The World Wide Web Consortium is the main international standards organization for the World Wide Web.

8.2 SPARQL Syntax

A generic SPARQL SELECT query has the following structure

8.2.1 Prefix Declaration

This part of the query defines prefixes assigned to the namespaces used in the query e.g.

PREFIX abc: <http://abc.com/resource/>

In the above example "abc" is the prefix for the namespace http://abc.com/resource/.

8.2.2 Result Clause

This section defines a set of all or selected variables required in response to a SPARQL query e.g.

SELECT {∗, data_1, data_2, ..., data_n}

The above example demonstrates a SELECT query that can ask for all variables in a result set using "*" or explicitly defined chosen variables.

8.2.3 Query Pattern

This section is the most important part of an effective query as it declares all the conditions to be matched for a said query. The syntax for a query pattern is as follows

WHERE {
...
Patterns to be matched
...
FILTER(... patterns to be filtered ...)
}

8.2.4 Query Modifiers

This section defines modifiers for the SPARQL query e.g. ORDER BY, LIMIT etc. The query modifiers help to sort (e.g. by relevance) or limit the number of results.

8.2.5 Example of a Complete SPARQL Query

Below is an example of a SPARQL query to find 30 books published in Cambridge

```
PREFIX bibo: <http://purl.org/ontology/bibo/>
PREFIX bio: <http://purl.org/vocab/bio/0.1/>
PREFIX blt: <http://www.bl.uk/schemas/bibliographic/blterms#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX event: <http://purl.org/NET/c4dm/event.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84\_pos#>
PREFIX isbd: <http://iflastandards.info/ns/isbd/elements/>
PREFIX org: <http://www.w3.org/ns/org#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdau: <http://rdaregistry.info/Elements/u/>
PREFIX madsrdf: <http://www.loc.gov/mads/rdf/v1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX void: <http://rdfs.org/ns/void#>
SELECT ?book ?isbn ?title WHERE {
   ?place rdfs:label "Cambridge" .
   ?publication event:place ?place.
   ?book
      blt:publication ?publication;
      bibo:isbn10 ?isbn;
      dct:title ?title.
}
LIMIT 30
```

8.3 Proposed System for Query Processing

Below are the main parts of SIRF that take part in query processing to translate a user query to a SPARQL query.

8.3.1 Query Optimiser

The Query Optimiser (as explained in Chapter 5 Section 5.3.3) does the initial query processing. The Query Handler module checks if there is some query already cached for the searched text. In case no match is found, it sends the query to the Syntax Analyser

(as discussed in Chapter 7 Section 7.3). Figure 8.1 recalls the work flow of the Syntax Analyser for natural language processing.

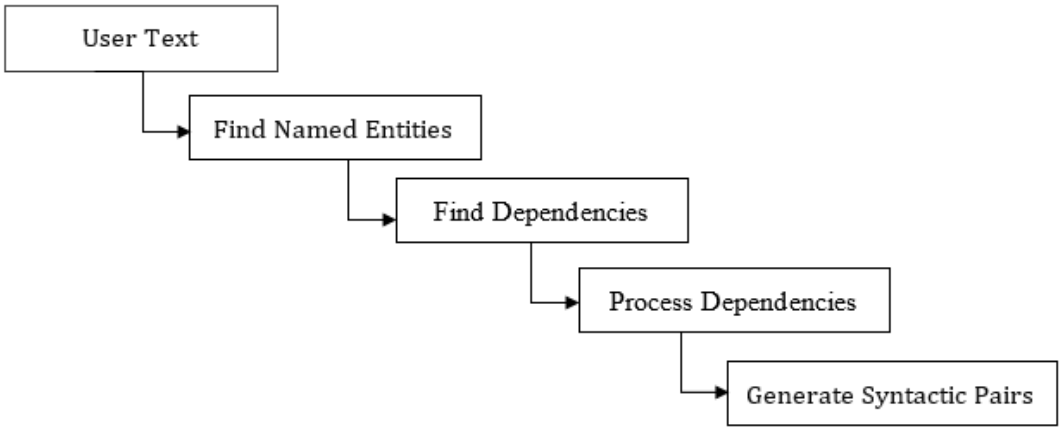


Figure 8.1: Syntax Analyser Work-flow (Source: Author, 2015)

The Syntax Analyser sends processed natural language statements to the Semantic Analyser that attaches CPI tags to the terms and dispatches these statements to the Query Formatter for further processing.

8.3.2 Query Formatter

The core function of the Query Formatter is to translate tagged concepts into a valid SPARQL query. Figure 8.2 shows the sub modules of the Query Formatter and Figure 8.3 describes the top-level work flow of the Query Formatter.

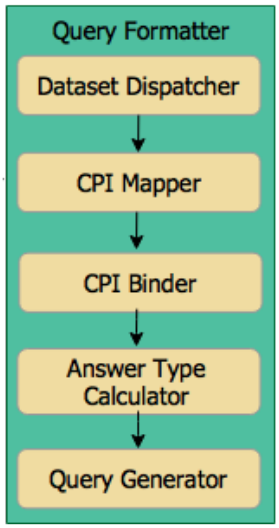


Figure 8.2: Query Formatter (Source: Author, 2015)

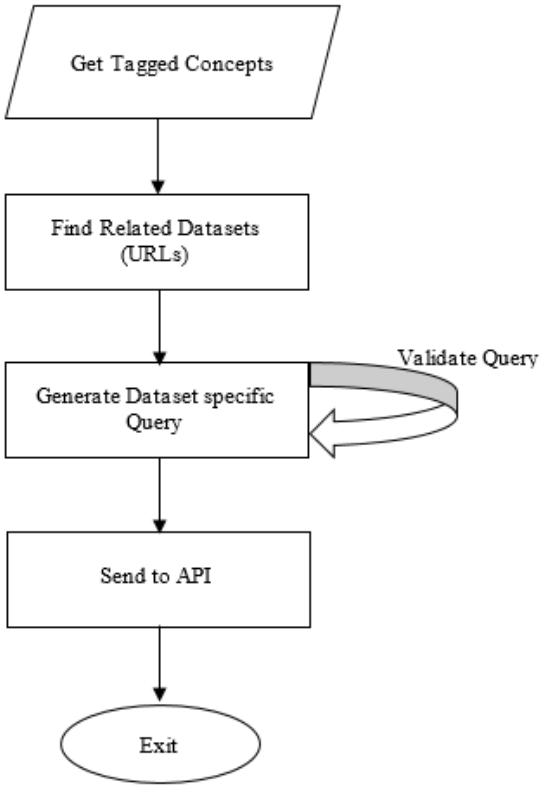


Figure 8.3: Query Formatter - Top level work flow (Source: Author, 2015)

The Query Formatter consists of five sub modules:

(i) Dataset Dispatcher (DSD) (see Section 8.4)

(ii) CPI Mapper (see Section 8.5)

(iii) CPI Binder (see Section 8.6)

(iv) Answer Type Calculator (ATC) (see Section 8.7)

(v) Query Generator (see Section 8.8)

8.4 Dataset Dispatcher (DSD)

The Query Formatter sends the tagged concepts to the Dataset Dispatcher that generates the list of all datasets that have matching concepts declared in the processed query. To produce a list of qualified datasets, the author proposes a formula as given below

$$\langle\ d \in \bigcap M\ \rangle \Leftrightarrow \langle\ \forall\ D \in M,\ d \in D\ \rangle$$

Where
M:= a set of all matched datasets
D:= A set of datasets for an individual concept
d:= An individual dataset

The above formula locates datasets for individual concepts and then find an intersection of the datasets to produce a list of qualified datasets (i.e. the datasets having all matching concepts). For example, for a query like "Simon's books", the Query Optimiser processes the set (simon, books) and generates the following output after mapping concepts

Instance⇒(foaf:Person, foaf:name, Simon),
Class⇒bibo:Book

From the above output, the Dataset Dispatcher extracts three main concepts i.e. foaf:Person, foaf:name and bibo:Book, and makes a list of datasets that have all three concepts. The process of finding matching concepts, is handled by the CPI Mapper as detailed below.

8.5 CPI Mapper

CPI Mapper matches the tagged concepts to the concepts defined in multiple datasets. Continuing the example from above (Section 8.4), the CPI Mapper takes the extracted concepts (i.e. foaf:Person, foaf:name and bibo:Book) from the Dataset Dispatcher and maps them to the cached datasets. Different datasets may use different ontologies to define similar concepts e.g. bibo:Book and dbpedia-owl:Book are two different classes but they refer to the same concept and some of the datasets may use bibo:Book while other may use dbpedia-owl:Book. The Ontology Cache of SIRF places similar concepts (from multiple datasets) in groups (as detailed in Chapter 6) that helps the CPI Mapper to map cross-domain concepts. Table 8.1 shows an example of multiple domain concept mapping.

Query: Simon's books Tagged Concepts ⇒ foaf:Person, foaf:name, bibo:Book	
Dataset	**Mapped Concepts**
British Library	foaf:Person, foaf:name, bibo:Book
DBPedia	dbpedia-owl:Person, foaf:name, dbpedia-owl:Book

Table 8.1: CPI Mapping (Source: Author, 2015)

8.6 CPI Binder

CPI Binder receives the list of qualified domains, list of processed statements and their mapped concepts from the Dataset Dispatcher. It processes all qualified domains in a serial fashion and for each domain it finds relations between the mapped concepts in three steps. In the first step it processes individual statements. In the second step it attempts to match missing links and in the third step it binds all statements together.

8.6.1 Step 1 - Find relations between concepts in syntactic pairs

This step involves finding relations between the concepts of a syntactic pair (from the Syntax Analyser). Algorithm 4 explains the process of linking concepts. The implementation of Algorithm 4 can be found in Appendix II (Listing 14).

Algorithm 4 Find Relations

Input:

statements : Mapped statements

Output:

Links : Mapped relations

while statements $\neq \emptyset$ **do**

for each statement $\in$ *statements* **do**

value1 := first index

value2 := second index

if CPI(value1) = 'property' and CPI(value2) = 'property' **then**

if value1→domain ∩ value2→domain **then**

Links ←(value1, value2, domain)

end if

else if CPI(value1) = 'property' and CPI(value2) = 'class' **then**

if value1→domain = value2 OR value1→range = value2 **then**

Links ←(value1, value2, domain/range)

end if

else if CPI(value1) = 'class' and CPI(value2) = 'class' **then**

p:= objectProperty

if (p→domain = value2 AND p→range = value1) OR (p→domain = value1 AND p→range = value2) **then**

Links←(value1, value2, p)

end if

else if CPI(value1) = 'class' and CPI(value2) = 'instance' **then**

if value2→class = value1 **then**

Links←(value1, value2, class)

end if

else if CPI(value1) = 'property' and CPI(value2) = 'instance' **then**

if value2→property = value1 **then**

Links←(value1, value2, property)

end if

Continue ...

Algorithm 4 Find Relations - Continued

```
    else if CPI(value1) = 'instance' and CPI(value2) = 'instance' then
      if (value2→property ∩ value1→property) OR (value2→class ∩ value1 →class)
then
        Links←(value1, value2, property/class)
      end if
    else if indirect_path(value1, value2) then
      Links←(value1, value2, indirect_path)
    else
      // no relation found
    end if
  end for
end while
```

Explanation of Algorithm 4:

In Algorithm 4, the author identifies six types of relationships between two concepts as explained below.

1. **Property-to-Property Relationship**
   This type of a relationship is formed when both concepts are identified as properties e.g. {author, name}. There are two types of properties defined in OWL [Antoniou and Van Harmelen, 2004] i.e. (i) Datatype Properties (that relate objects to data type values e.g. name, gender etc.) and (ii) Object Properties (that relate objects to other objects e.g. author, taughtBy etc.). A property can be associated to a class (e.g. property foaf:name belongs to class foaf:Person) or it can have associated domain[52] and range[53] (e.g. for a relation "bibo:Book dct:creator foaf:Person", bibo:Book is the domain of dct:creator and foaf:Person is the range of dct:creator). Table 8.2 shows a list of identified property to property relations.

[52]The domain of an OWL property is the subject resource of that property.
[53]The range of an OWL property is the object resource of that property.

Property 1	**Property 2**	**Related**
Datatype	Datatype	if both properties belong to same class
Datatype	Object	if class of Property 1 is equal to the domain or range of Property 2
Object	Object	if both properties have same domain OR domains or ranges of both properties are related

Table 8.2: Property-to-Property Relations (Source: Author, 2015)

2. **Property-to-Class Relationship**
   A property P and a class C are related if P is a datatype property and class of P is equal to C e.g. {Person, name}
   OR
   P is an object property and, domain or range of P is equal to C e.g. {Person, author}

3. **Property-to-Instance Relationship**
   A property P and an instance I are related if
   property of I is equal to P e.g.{title, Hamlet(a book title)}
   OR
   class of I is equal to domain or range of P e.g. {author, Hamlet}

4. **Class-to-Class Relationship**
   A class is related to another class if both classes are linked by a property e.g. {Journal, Person} where class Journal can be linked to class Person with a property author i.e. Journal has an author and author is a Person.

5. **Class-to-Instance Relationship**
   A class C is related to an instance I if
   class of I is equal to C e.g. {Book, Da Vinci} where class Book has an instance Da Vinci
   OR
   class of I is related to C e.g. {Neilson, Books} where class of *Neilson* is Person that is related to class Book.

6. **Instance-to-Instance Relationship**
   An instance is related to another instance if their classes are related e.g. {Cambridge, UK} where class of *Cambridge* is City and class of *UK* is Country.

8.6.2 Step 2 - Find derived and missing relations

In the event, there are more than one processed statements, received from the Dataset Dispatcher, the CPI Binder processes all statements to find possible links among them. In case there is no direct link found between two statements, the CPI Binder searches for any indirect path that can be a possible link e.g. Book and publication date can be linked by an indirect path such that {Book has a publication event} and {publication event has a publication date}. Different domains may have different indirect paths between two concepts. SIRF precomputes indirect paths from different domains and places them in the Ontology Cache as shown in Figure 8.4. The prototype for computing indirect paths can be found in Appendix II (Listing 10).

first_node	last_node	links	domain_id
http://purl.org/ontology/bibo/Book	http://xmlns.com/foaf/0.1/Agent	{http://purl.org/ontology/bibo/Book,dct:creator,ht...	1
http://purl.org/ontology/bibo/Book	http://xmlns.com/foaf/0.1/Person	{http://purl.org/ontology/bibo/Book,dct:creator,ht...	1
http://purl.org/ontology/bibo/Book	http://purl.org/dc/terms/Agent	{http://purl.org/ontology/bibo/Book,dct:creator,ht...	1
http://purl.org/ontology/bibo/Book	http://purl.org/vocab/bio/0.1/Birth	{http://purl.org/ontology/bibo/Book,dct:creator,ht...	1
http://purl.org/ontology/bibo/Book	http://purl.org/vocab/bio/0.1/Death	{http://purl.org/ontology/bibo/Book,dct:creator,ht...	1
http://purl.org/ontology/bibo/Book	http://www.bl.uk/schemas/bibliographic/blterms#PublicationEvent	{http://purl.org/ontology/bibo/Book,blt:publicatio...	1
http://purl.org/ontology/bibo/Book	http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing	{http://purl.org/ontology/bibo/Book,blt:publicatio...	1
http://purl.org/ontology/bibo/Book	http://purl.org/dc/terms/Location	{http://purl.org/ontology/bibo/Book,blt:publicatio...	1
http://purl.org/ontology/bibo/Book	http://reference.data.gov.uk/def/intervals/CalendarYear	{http://purl.org/ontology/bibo/Book,blt:publicatio...	1
http://purl.org/ontology/bibo/Book	http://www.bl.uk/schemas/bibliographic/blterms#TopicDDC	{http://purl.org/ontology/bibo/Book,dct:subject,ht...	1
http://purl.org/ontology/bibo/Book	http://www.w3.org/2004/02/skos/core#Concept	{http://purl.org/ontology/bibo/Book,dct:subject,ht...	1

Figure 8.4: Indirect Paths - The Ontology Cache (Source: Author, 2015)

Apart from the direct and indirect links, the CPI Binder also tries to find missing links that may connect two statements. For example, for a query "list names of some authors", the term *author* may produce multiple interpretations (as shown in Figure 8.5 and the term *names* may produce a single interpretation (as shown in Figure 8.6).

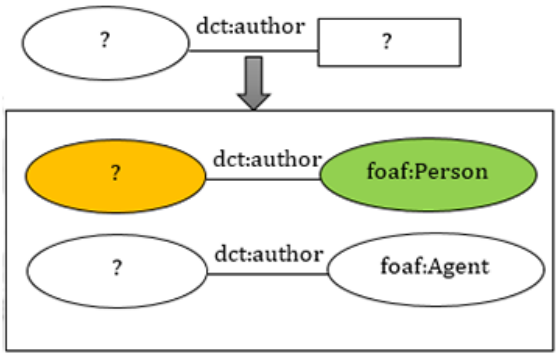


Figure 8.5: Multiple CPI Relations (Source: Author, 2015)

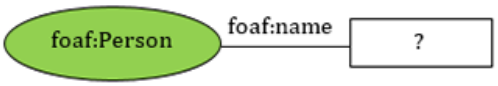


Figure 8.6: Single CPI Relation (Source: Author, 2015)

The CPI Binder will link both concepts using their common concept (in this case foaf:Person) as shown in Figure 8.7.

Figure 8.7: Linking the missing relations (Source: Author, 2015)

8.6.3 Step 3 - Bind relations

This step binds related statements. The statements that are linked together are classified as qualified statements. Only qualified statements take part in the process of query formation.

8.6.4 Examples

Below are some examples of finding concepts relations (using the British Library dataset)

A. Example-I

	Query: List all authors born in 1945
Dependency Sets	(authors, born) in(born, 1945)
CPI Tags	(dct:creator{property}, bio:event{property}) (bio:event{property}, (bio:Event, bio:date,1945){instance})
CPI Binding	**Step-1:** (? dct:creator ?) (? bio:event bio:Event) (bio:Event, bio:date,1945) **Step-2:** (? dct:creator foaf:Person) (? dct:creator foaf:Agent) (foaf:Person bio:event bio:Event) (bio:Event, bio:date,1945) **Step-3:** (? dct:creator foaf:Person) (foaf:Person bio:event bio:Event) (bio:Event, bio:date,1945)

Table 8.3: CPI Binding - Example I (Source: Author, 2015)

B. Example-II

Query: What is the name of the author of Da Vinci?	
Dependency Sets	(name, author) (author, Da Vinci)
CPI Tags	(foaf:name{property}, dct:creator{property}) (dct:creator{property}, (bibo:Book, dct:title, Da Vinci){instance})
CPI Binding	**Step-1:** (? foaf:name ?) (? dct:creator ?) (bibo:Book dct:title Da Vinci) **Step-2:** (foaf:Person foaf:name ?) (? dct:creator foaf:Person) (? dct:creator foaf:Agent) (bibo:Book dct:title Da Vinci) **Step-3:** (bibo:Book dct:creator foaf:Person) (foaf:Person foaf:name ?) (bibo:Book dct:title Da Vinci)

Table 8.4: CPI Binding - Example II (Source: Author, 2015)

Figure 8.8 further elaborates Example II giving a visual representation of finding qualified links.

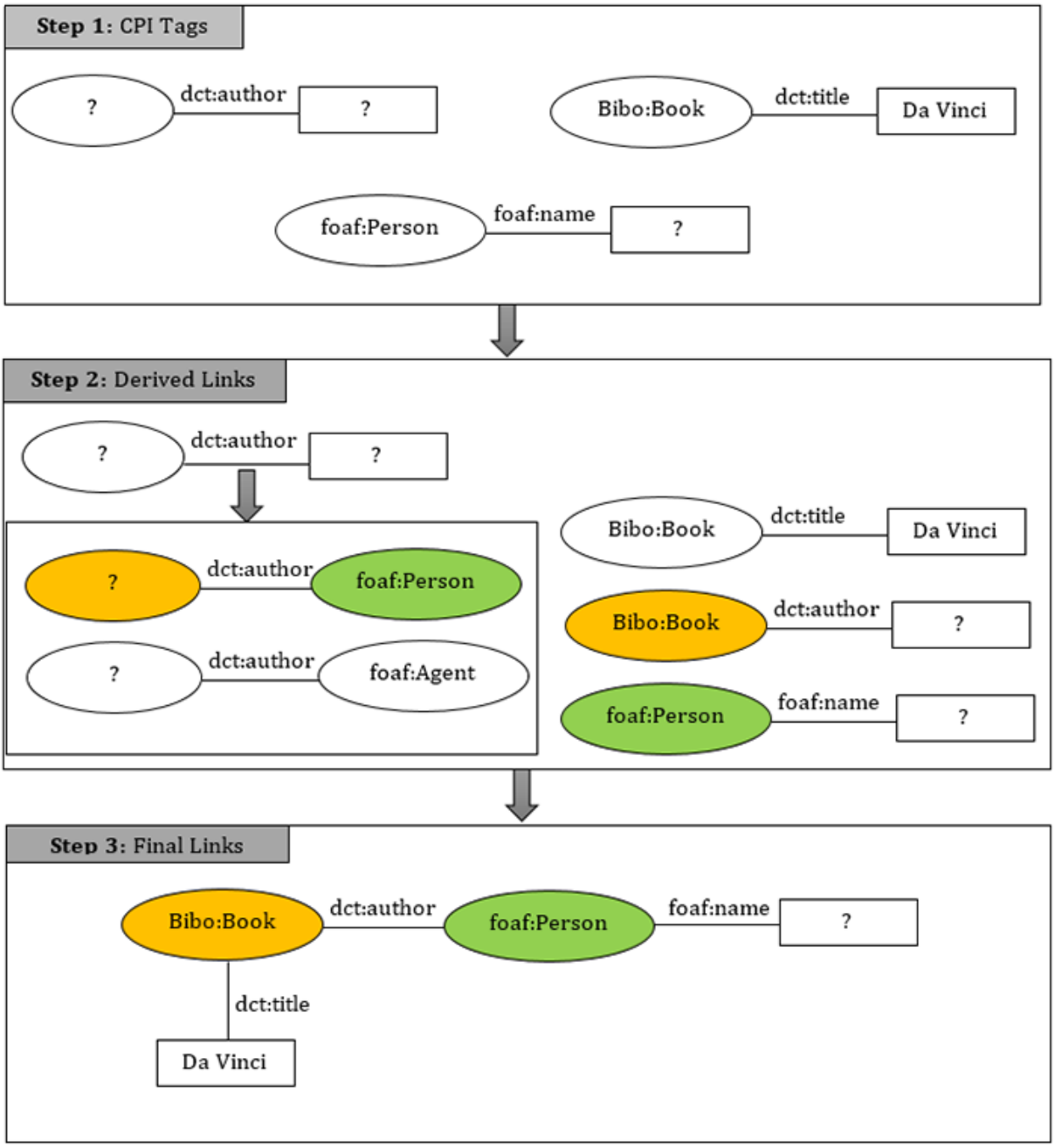


Figure 8.8: CPI Binding (Source: Author, 2015)

8.7 ATC (Answer Type Calculator)

The Answer Type Calculator (ATC) module processes all candidate concepts (generated by CPI Binder) and finds a category for the result called Calculated Answer Type (CAT). ATC uses its own grammar rules to annotate CAT for given query concepts. Algorithm 5 explains the process of finding CAT value. The implementation of Algorithm 5 can be found in Appendix II (Listing 13).

Algorithm 5 Calculate CAT

Input:
Q : set of candidate concepts
D : set of declared concepts
Output:
CAT : calculated answer type

```
L ← findMissingLinks(Q)
if L ≠ ∅ then
  if (p ∈ D) and (p → type is property) then
    if (p→type is "object_type") and (p→range = ∅) then
      range = p→findRange()
      CAT ← range
    else if (p→type is "data_type") and (p→value = ∅) then
      CAT ← p
    end if
  end if
else if L = ∅ then
  List_class ← getClasses(D)
  List_prop ← getProperties(D)
  if List_class ≠ ∅ then
    CAT ← List_class
  else if (List_class = ∅) and (List_prop ≠ ∅) then
    CAT ← List_prop
  end if
end if
```

Explanation of Algorithm 5:

Algorithm 5 (as described above) attempts to find the intent of a user query by calculating answer type of the query. The algorithm computes missing links in a query. In the case a missing link found, it checks the type of the missing link. If the missing link is a data type property then the property will be assigned to CAT e.g. for a query "country names", the missing link is the value of *names* (which is a data type property) and the CAT will be *names*. If the missing link is an object type property and range of the property is not known, the CAT will be the range of that property e.g. for a query "list 50 authors", *authors* is an object type property and its range is not known in the query hence the range of the property *Person* will be the calculated answer type. In the case there are no missing links found un the query, the algorithm will locate defined concepts in the query i.e. classes and properties. If some class concepts found in the query, they are assigned

to the CAT. Table 8.5 presents some examples of calculated answer types.

User Query	CAT
List all authors born in 1945	Person (range of authors)
Find books published in New York	Book (class)
search for a book with ISBN 9780729408745	Book (class)
Which states border Texas	State (class)
Horror movies	Movie(class)
Find English songs	Song (class)
population of cities in California	population (data type property)
which city is the largest city in Pakistan	City (class)

Table 8.5: Examples of Calculated Answer Type (Source: Author, 2015)

8.8 Query Generator

Manipulating qualified statements from CPI Binder, the Query Generator formulates a SPARQL query. The process of query formation includes generation of result clause, query patterns, filters and query modifiers. In the following sections, the author explains the rules followed by the Query Generator to handle conditional queries (Section 8.8.1), statement mapping to formulate SPARQL query(Section 8.8.2) and a query formation algorithm (Section 8.8.3).

8.8.1 Rules for Conditional Queries

8.8.1.1 Negation

The Query Generator handles negation in two ways as given below.

A. **Using Logical NOT (!)**
To apply negation to an instance or Named Entity, the Query Generator uses logical not (!) operator with a regular expression in FILTER clause. Below is an example of a query (using logical not (!)) that finds all persons whose family name is not 'hardy'.

```
SELECT * WHERE {
    {?Person a <http://xmlns.com/foaf/0.1/Person>}
    {?Person <http://xmlns.com/foaf/0.1/givenName> ?given_name}
    {?Person <http://xmlns.com/foaf/0.1/familyName> ?family_name}
```

```
FILTER(!REGEX(str(?family_name),"hardy","i"))
}
```

B. **Using NOT EXISTS**

To apply negation to a class or property concept, the Query Generator utilises NOT EXISTS keyword in FILTER clause. Below is an example of a query (using NOT EXISTS) that finds all the persons who are not dead.

```
SELECT * WHERE {
    {?Person a <http://xmlns.com/foaf/0.1/Person>}
    {?Person <http://xmlns.com/foaf/0.1/givenName> ?given_name}
    {?Person <http://xmlns.com/foaf/0.1/familyName> ?family_name}
FILTER(NOT EXISTS{?Person <http://dbpedia.org/ontology/deathDate> ?
   date})
}
```

8.8.1.2 Conjunction

Conjunction in this context, is used to build a relation between two syntactic pairs connected by coordinating conjunctions such as 'and' or 'or' e.g. "find person whose name is john or jack". The Query Generator receives syntactic pairs with related conjunctions and processes them using logical OR (||) and logical AND (&&) operators in SPARQL. For the given query, the Query Generator receives syntactic pairs with related conjunctions [e.g. or((name, john), (name, jack))] and produces the following SPARQL query.

```
SELECT * WHERE {
   {?Person a <http://xmlns.com/foaf/0.1/Person>}
   {?Person <http://xmlns.com/foaf/0.1/name> ?name}
   FILTER(REGEX(?name, "jack", "i") || REGEX(?name, "john", "i"))
}
```

8.8.1.3 Numeric Quantifiers

The numeric quantifiers are the numeric quantities specified for a concept or noun. If numeric quantifiers are attached to a class or property, the Query Generator attempts to solve it using the LIMIT keyword defined in SPARQL e.g. for a query "list 50 persons born in 1972" the Query Generator will generate following SPARQL query.

```
SELECT * WHERE {
```

```
   {?Person a <http://xmlns.com/foaf/0.1/Person>}
   {?Person <http://xmlns.com/foaf/0.1/name> ?name}
   {?Person <http://dbpedia.org/ontology/birthDate> ?date}
   FILTER(?date = "1972"^^<http://www.w3.org/2001/XMLSchema#gYear>)
}LIMIT 50
```

If the numeric quantifier is attached to an instance variable, the Query Generator will solve it using comparison operators i.e. equal to (=), greater than (>), greater than and equal to (>=), less than (<) and less than and equal to (<=). For example for a query "list upto 50 persons whose age is above 50", the Query Generator will generate following query.

```
SELECT * WHERE {
   {?Person a <http://xmlns.com/foaf/0.1/Person>}
   {?Person <http://xmlns.com/foaf/0.1/name> ?name}
   {?Person <http://dbpedia.org/ontology/age> ?age}
   FILTER(?age > 50)
}LIMIT 50
```

8.8.2 Statement Mapping

8.8.2.1 Result Clause

The result clause is generated as "SELECT * " where "*" makes sure to return all variables defined in the query.

8.8.2.2 Query Patterns

Table 8.6 displays the query patterns generated in SPARQL syntax for qualified query concepts.

Qualified Concepts	**SPARQL syntax**
Classes	{?class_variable a <class_uri>}
Data type property	{?class_variable <property_uri> ?property_var}
Object type property	{?class1 <property_uri> ?class2}
Instance	{?class_variable a <class_uri>} {?class_variable <property_uri> ?instance_variable}
Multiple CPI relations	Concatenated by OPTIONAL operator
Filters for same instance	Concatenated by || (OR) operator
Filters for multiple instances	Concatenated by && (AND) operator

Table 8.6: Mapped Query Elements (Source: Author, 2015)

8.8.3 Algorithm for query formation

This section describes the algorithm for query formation. The generated query is sent to the API module that runs it on a related SPARQL endpoint and then fetches results (if available), as shown in Algorithm 6 below.

Algorithm 6 Generate SPARQL Query

Input:

D : set of declared statements

ds : dataset URL

Output:

Q : SPARQL Query

Declare:

S : Query Statements

statements(class, property, link) ← parse(D)

cat = getCAT() // from Algorithm 5

schema = getSchema(cat)

S→statements = schema→Statements

Continue ...

Algorithm 6 Generate SPARQL Query - Continued

```
for each statement s ∈ D do
  if (s→type = 'class') then
    S→statement = createClassStatement()
  end if
  if (S→type = 'property') then
    S→statement = createPropertyStatement()
  end if
  if (S→type = 'instance') then
    S→statement = createInstanceStatement()
    S→filters = createInstanceFilter()
  end if
end for
if (D→quantification ≠ ∅) then
  if (D→quantification→concept = 'cat') then
    createLimit(D→quantification→num)
  else if (D→quantification→concept ≠ 'class' AND D→quantification→concept ≠
'property') then
    createFilterCondition()
  end if
end if
if (D→conjunction ≠ ∅) then
  createConjuctions()
  createFilterConditions()
end if
if (D→negation ≠ ∅) then
  if (D→negation→concept = instance) then
    createNotEqualFilter()
  else
    createNotExistFilter()
  end if
end if
Q→createSelectStatements()
Q→joinStatements(S)
Q→addLimit()
return Q
```

Explanation of Algorithm 6:

The algorithm formulates a SPARQL query for a particular dataset using parsed statements from CPI Binder and CAT value from the Answer Type Calculator. In the first step, the algorithm generates schema statements based on CAT value. The schema statements are used for result optimisation (see details in Chapter 9). In the second step, the algorithm generates statements for pattern matching. For each given statement (from CPI Binder), the algorithm generates patterns for given concepts (i.e. class, property and instance). For an instance, the algorithm generates two pattern statements (i) statement for the class and property of instance and (ii) filter statement. In the third step, the algorithm generates statements to handle conditions for quantification, conjunction and negation. In the final step, the algorithm joins all statements and formulates a query. The implementation of Algorithm 6 can be found in Appendix II (Listing 20).

8.9 Working Example for Query Generation

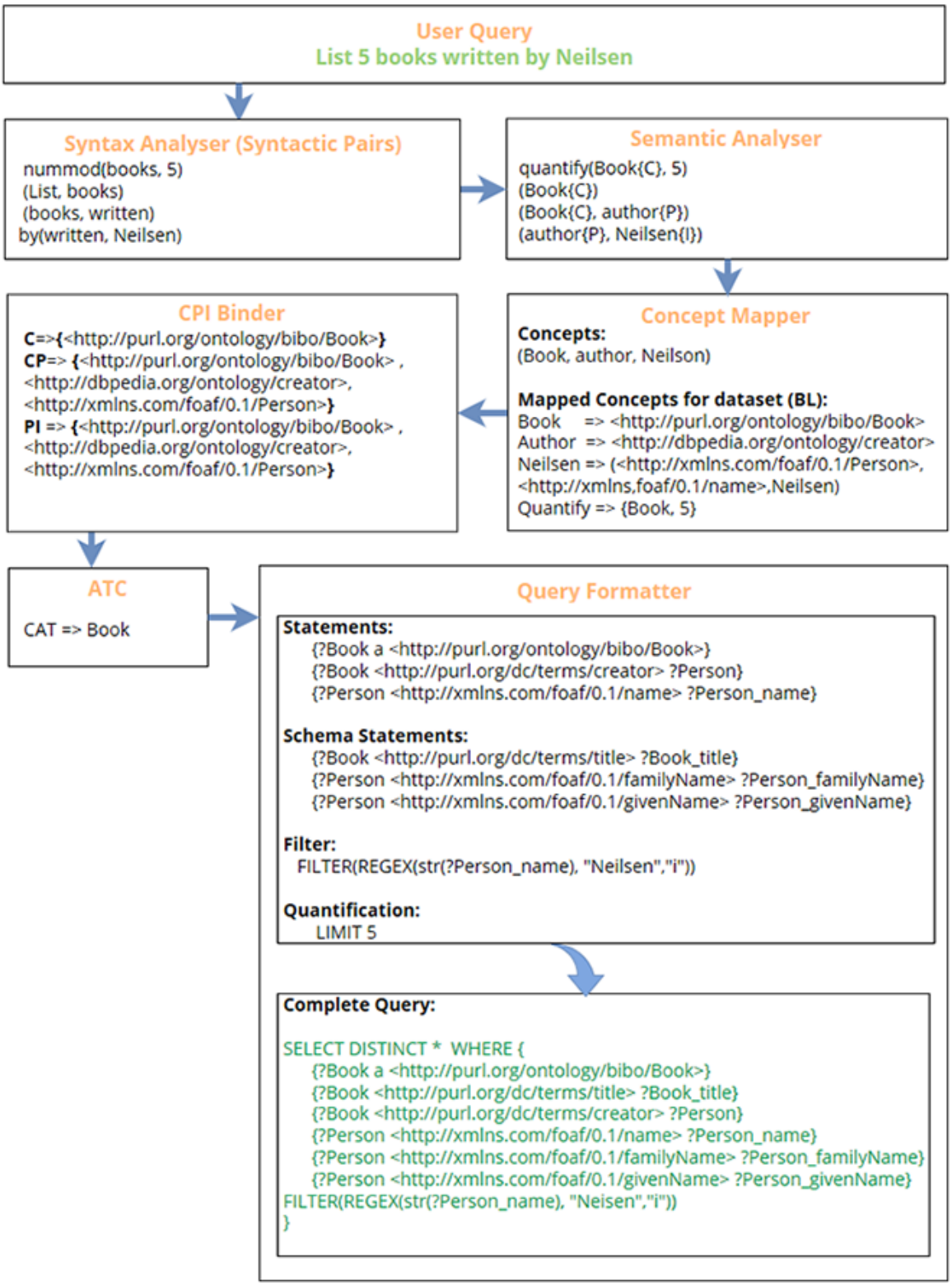


Figure 8.9: Working Example for Query Generation (Source: Author, 2015)

8.10 Running SPARQL Queries

The API module of SIRF, receives a SPARQL query from the Query Generator and sends query requests to related SPARQL endpoints. Initially, the author tested federated SPARQL queries[54] to fetch data from heterogenous datasets. However, this approach was found limiting the performance of the system i.e.

- It is time consuming as it fetches all data in one go and then returns results
- If any SPARQL end point gives an error, results are not returned
- It is difficult to identify result set and apply result schemas

The API sends query requests in a serial fashion using the order of ranking (explained in Chapter 9).

8.11 Summary

Figure 8.10 describes an overview of the SPARQL query creation work-flow. This chapter has explained the process of query formation which involves a number of SIRF's modules. The Query Optimiser and the Query Formatter modules map the concepts from a user query to a semantic query and convert the mapped relations into SPARQL syntax. The API processes queries on related endpoints that in response return data (if available).

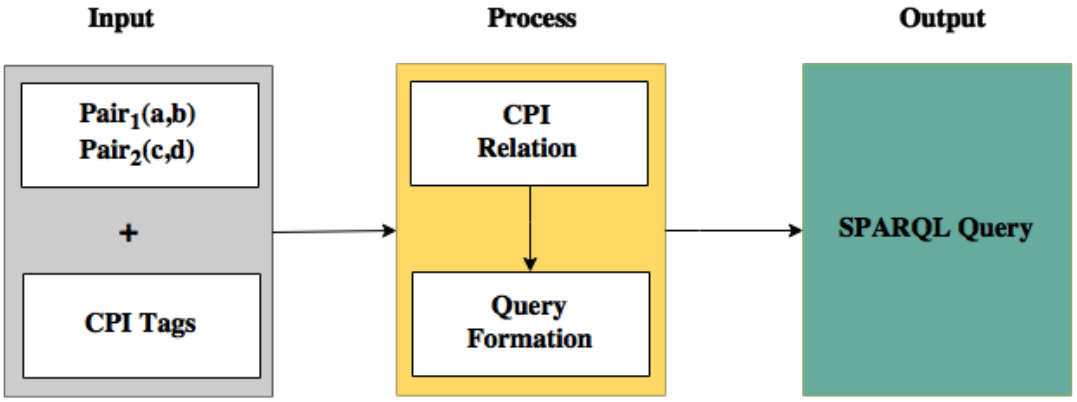


Figure 8.10: An overview of Chapter 8 (Source: Author, 2015)

The response from the SPARQL endpoints is sent to the Result Optimiser for further action. The next chapter explains the process of result optimisation.

[54] A federated query is an extension of SPARQL query for executing queries distributed over different SPARQL endpoints using the SERVICE keyword.

Chapter 9

Result Optimisation

9.1 Introduction

There are a number of challenges faced by any semantic search tool e.g. translating a user query to a formal SPARQL query, scalability, guidance and optimising query results for better readability and visualisation etc. In the previous chapters, the researcher discussed other modules of SIRF which processed information and converted a user query to a SPARQL query. The focus of this chapter is the result optimisation for the data received as a result of running queries over different SPARQL end-points. Most of this chapter has been taken from the author's published work [Fatima *et al.*, 2015b]. The chapter highlights two aspects of the result optimisation i.e. (i) result ranking and (ii) result readability. This chapter also introduces an algorithm for result ranking and suggests a layout schema that defines the result template for the user interface.

This chapter has been divided in two main parts. The first part (Section 9.2) explains the concept of result optimization elaborating the requirement of result ranking and using result schemas for a better user interface. The second part (Section 9.3) explains the Result Optimiser module of the proposed framework to optimise results returned from SPARQL endpoints.

9.2 What Optimisation?

One important feature for user friendly search interfaces is 'Result Optimisation'. The size of semantic data is increasing constantly and this increase adds a need for procedures to rank results and display the results in a user friendly way. Existing semantic search tools (as discussed in related work Chapter 4) now include large repositories of ontology

data sets but none of them yet handles result optimisation. The author considers two basic aspects of result optimisation as explained below.

9.2.1 Result Ranking

Search result ranking is a measure of relative importance of web pages so that users quickly make sense of the vast heterogeneity of the World Wide Web [Page *et al.*, 1999]. The main target of the search result ranking is to get more relevant and reliable results on top. There are various result ranking techniques used by conventional syntax-based search engines i.e. Pagerank [Langville *et al.*, 2008], adaptations of vector machines [Chapelle and Keerthi, 2010], neural network approach [Burges *et al.*, 2005] and gradient boosted regression trees (GBRT) [Zheng *et al.*, 2008]. The above search ranking techniques fit well to rank web pages. Whilst the semantic search targets linked semantic data rather than web pages, it requires different ranking schemes.

Ranking techniques have also been applied in traditional ontology engineering [Hotho *et al.*, 2006]. The author's proposed ranking solution is influenced by Pagerank algorithm [Page *et al.*, 1999] and OntoKhoj [Patel *et al.*, 2003] where both measure the importance of a page or concepts by linked hyperlinks. [García *et al.*, 2013] introduced ranking based on preference model. The author approaches this problem by calculating frequency of concept usage.

Since semantic web data is based on ontologies, the ranking of the search results must be somehow ordered on the basis of semantic relevancy. Due to different interests and points of view, many people define their own ontologies for the same domain of interest e.g. the concept 'Person' has been defined in various namespaces i.e.

http://xmlns.com/foaf/0.1/Person
http://schema.org/Person
http://www.w3.org/ns/person#Person.

The flexibility of defining different ontologies for the same concepts raises a challenge of ontology alignment and trust levels (i.e. which ontologies are the most trusted ones). Whilst ontology alignment is outside the scope of this work, the proposed system intends to utilise existing tools for ontology alignment. The Result Optimiser ranks results in two steps. In the first step, it finds a list of all namespaces that define the required concept e.g. all namespaces that define the concept Person. The list of namespaces is ranked based on popularity. The Ontology Cache (as defined in Chapters 5 and 6) saves the list of namespaces and their frequency of usage. Table 9.1 shows an example of ranking

namespaces for the concept 'Person'. The example explains that the FOAF namespace (http://xmlns.com/foaf/0.1/) has the top rank for the concept 'Person' because it is the most used namespace on different domains for this concept.

Concept URI	**Rank**
http://xmlns.com/foaf/0.1/Person	1
http://schema.org/Person	2
http://www.w3.org/ns/person#Person	3
http://dbpedia.org/ontology/Person	4
http://www.bbc.co.uk/ontologies/coreconcepts/Person	5
...	...

Table 9.1: Namespace Ranking (Source: [Fatima *et al.*, 2015b])

Similarly, Table 9.2 displays an example of property ranking. Whilst the same property can be used with multiple classes in different namespaces, it is required to set some priority ranking for each combination of class and the property. From the given example (Table 9.2), the property http://xmlns.com/foaf/0.1/name has been used with multiple classes e.g. 'Person', 'Organisation', 'Group', 'Document' etc. The proposed system saves the counters for each set of property and class and ranks the highest counter on top.

The property ranking helps the search tool to rank results especially when there are no concepts identified from a user query e.g. for a search query 'Milli', there are no specified classes or concepts and the keyword ('Milli') is mapped to the property foaf:name (http://xmlns.com/foaf/0.1/name). The property foaf:name can be linked to multiple classes e.g. name of a 'Person', name of an 'Organisation' etc. In such a case the search system will find results in an order based on defined ranks. For the above example the proposed system will display names of the 'Persons' matched with the keyword on the top followed by the names of the organisations and so on.

Property URI	**Class**	**Rank**
http://xmlns.com/foaf/0.1/name	foaf:Person	1
http://xmlns.com/foaf/0.1/name	foaf:Organization	2
http://xmlns.com/foaf/0.1/name	foaf:Group	3
http://xmlns.com/foaf/0.1/name	foaf:Document	4
...	...	...

Table 9.2: Property Ranking (Source: [Fatima *et al.*, 2015b])

In the second step, the Result Optimiser finds and ranks the domains for each namespace. Table 9.3 shows a list of ranked domains for the concept 'Person' from FOAF namespace. The domains are ranked based on the instance count for a particular concept on a given domain. The example (from Table 9.3) shows that the Freebase domain has the highest number of instances for the concept 'Person'.

Domain	Instance count	SPARQL endpoint	Rank
Freebase	3,401,174	http://www.freebase.com/base/sparql	1
DBPedia	1,450,000	http://dbpedia.org/snorql	2
British Library	1,211,026 (source: SPARQL end-point)	http://bnb.data.bl.uk/sparql	3
...	...	...	...

Table 9.3: Domain Ranking (Source: [Fatima *et al.*, 2015b])

9.2.2 Readability

Recalling from Chapter 4, the existing semantic search tools lack readability of results. The proposed system introduces a result schema. The schema defines key attributes for each class in an ontology with a pre-defined layout (as defined in Chapter 5).
The Query Formatter adds schema attributes to the result clause while generating SPARQL queries e.g. for a user query that searches 'Persons', the schema attributes defined are name, givenName and familyName and the SPARQL statements generated are

SELECT ?person ?name ?givenName ?familyName
?person a foaf:Person
?person foaf:name ?name
?person foaf:givenName ?givenName
?person foaf:familyName ?familyName

9.3 Result Optimiser Module

SIRF introduces the Result Optimiser module that aims to optimise results fetched from different SPARQL end-points. The Result Optimiser module has been sub-divided in two main parts i.e. (i) Result Ranker and (ii) Schema Dispatcher.

9.3.1 Result Ranker

In a traditional search environment, the results are ranked based on various factors i.e. the relevancy, popularity and customisation. Since the semantic-based search works on a different data structure, it needs different ranking techniques to achieve the similar custom. There have been a number of ranking algorithms introduced for semantic search i.e. SemRank (a relation based ranking) [Anyanwu *et al.*, 2005], a page rank algorithm by [Vijayadeepa and Ghosh, 2013], an entity based ranking approach by [Wei *et al.*, 2011] etc. [Jindal *et al.*, 2014] divide semantic search ranking in three stages (i) entity-based ranking, (ii) relationship ranking and (iii) semantic document ranking.

In the present work, the author uses a ranking technique, as a part of the result optimisation system, that is based on semantic classification. The author proposes an algorithm to rank ontology concepts to arrange results in an order of top ranked ontology.

Result Ranker works in combination with the Query Formatter during the query generation process. Algorithm 7 explains the procedure of saving concepts ranks while parsing ontologies (as described in Chapter 6). Algorithm 8 explains the procedure of ranking concepts and domains while generating SPARQL queries (as explained in Chapter 8).

9.3.1.1 Algorithm to Save Ranks

Algorithm 7 Save Ranks

```
Input:
    ontology: ontology to parse
Declare:
    L: list of cached concepts
    D: list of cached domains
    G: concept Groups
concepts ← parse(ontology)
domain ← getDomain(ontology)
for each entity c ∈ concepts do
  if c ∈ L then
    if domain ∉ D then
      cacheDomain()
      increment_concept_counter()
    else
      //ignore, do nothing
    end if
  else
    if (c ∈ G) then
      c→assignGroup()
    else
      c→assignNewGroup()
    end if
    if (domain ∉ D) then
      cacheDomain()
    end if
    initialise_concept_counter()
  end if
end for
```

Explanation of Algorithm 7:

Algorithm 7 works as a part of ontology caching process (Chapter 6 Section 6.4). It grabs ontology concepts (i.e. classes and properties) from a dataset and checks if the concept has already been recorded in the Ontology Cache. If the concept exists but domain is not listed, the system caches the domain and increments the usage counter for the given concept. The greater is the frequency of usage counter, the higher will be the concept

ranking. If the concept does not exist in the Ontology Cache, the algorithm tries to align it with the existing concepts and assign a matching group code. In the case the concept is not aligned to any cached concept, the system creates a new group, assigns the concept to that group and initialises the concept's usage counter with 1. The implementation of Algorithm 7 can be found in Appendix I (Listing 4).

9.3.1.2 Algorithm to Get Ranks

Algorithm 8 Get Ranks

Input:
concepts: concepts derived from user query
Output:
ranks: list of ranked domains

```
concept ← findCAT(concepts) //calculate answer type
namespaces ← getNameSpaces(concept)
namespaces→sortByFrequency()
for each namespace ns ∈ namespaces do
  domains ← getDomains(ns)
  domains→sortByConceptCount()
  for each domain d ∈ domains do
    ranks ← d
  end for
end for
return ranks
```

Explanation of Algorithm 8:
Algorithm 8 presents an overall view of generating a list of ranked domains. The algorithm receives all concepts tagged in a search query from the Query Formatter and tries to find a CAT value (using Algorithm 5 (Calculate CAT) from Chapter 8) that serves as the main concept. In the first step, the algorithm fetches all the namespaces (from the Ontology Cache) that define the given concept. In the second step, it extracts all domains that use the ranked namespaces. The ranking for the domains is calculated by the number of instances each domain retains for a given concept. The implementation of Algorithm 8 can be found in Appendix II (Listing 20).

9.3.2 Schema Dispatcher

For better visualisation and readability, the results received from SPARQL endpoints need to be re-formatted. The proposed system introduces defining result schema to build the result display. It saves basic schema for semantic concepts. Table 9.4 shows some example schemas. Schema Dispatcher gets schema attributes for a given concept. Schema attributes are used as a result set or part of a result set for a SPARQL query. For example for a query "Book", there are no attributes explicitly defined, in such a case the Schema Dispatcher attaches a basic schema that is "book title" and adds it to the result set. The pre-defined schema is used to set up the result layout for better readability.

Class	**Attributes**	**Schema layout**
http://xmlns.com/foaf/0.1/Person	name, givenName, familyName	name(familyName, givenName)
http://purl.org/ontology/bibo/Book	title	title
...	...	...

Table 9.4: Schema Examples (Source: [Fatima *et al.*, 2015b])

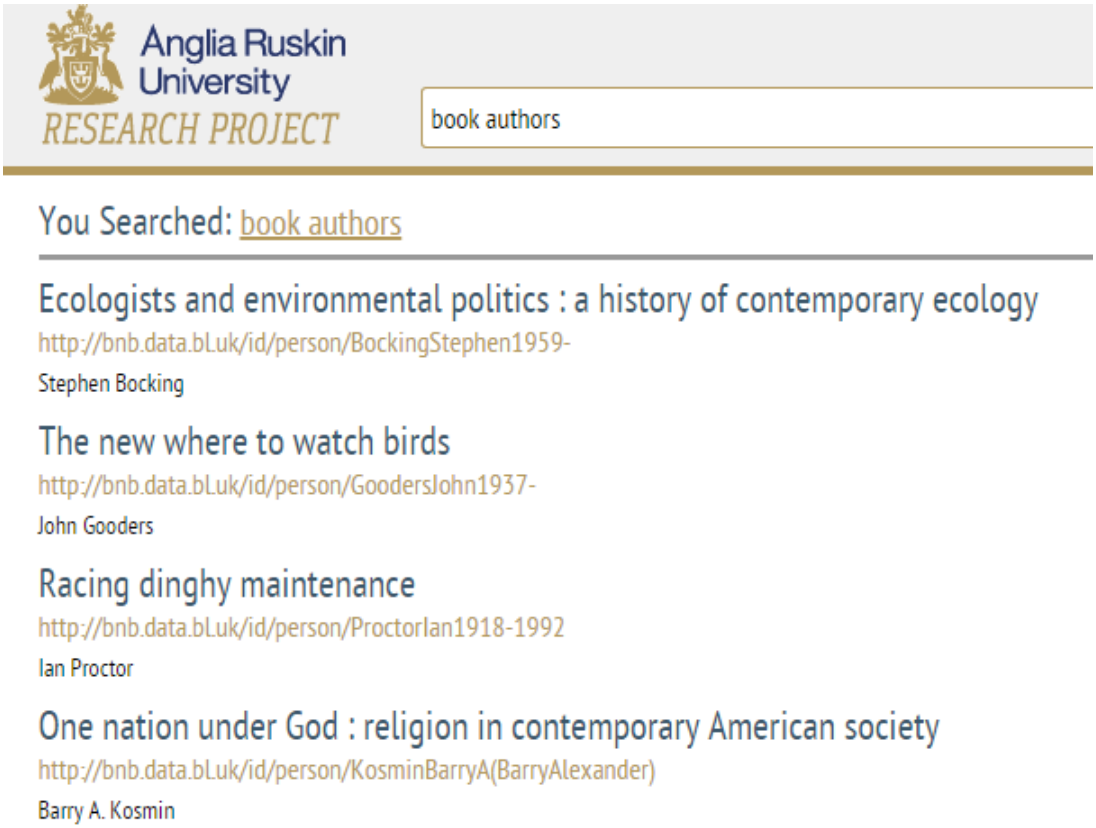


Figure 9.1: An example of schema based presentation of results (Source: Author, 2015)

9.4 Summary

Among various challenges for creating an effective semantic search tool is the question of how to optimise query results (result optimisation). Result optimisation from the perspective of usability involves result ranking and result layout for better visualisation. SPARQL endpoints are the access points for RDF datasets. Different domains use different namespaces for similar concepts that raises the challenge of ontology alignment and result ranking. This chapter has focused on those aspects of result optimisation that will improve usability. The proposed system ranks results based on the popularity of namespaces and the number of instances that a domain possesses for a particular concept. This chapter has also described a layout schema for the captured results.

The next chapter explains the construction of a prototype developed by the author (which is based on the architecture of SIRF). The chapter also provides an evaluation of the framework.

Chapter 10

Prototype and Evaluation

10.1 Introduction

"Design science consists of two basic activities, build and evaluate. These parallel the discovery justification pair from natural science. Building is the process of constructing an artifact for a specific purpose; evaluation is the process of determining how well the artifact performs."[March and Smith, 1995]. Following the above definition (by March and Smith) and adopting the Design Research methodology proposed by [Blessing and Chakrabarti, 2009] (as discussed in Chapter 3), the author developed a prototype and evaluated its effectiveness against the criteria proposed (as discussed in Chapter 4.

This chapter describes the construction and testing of a prototype to evaluate the performance of SIRF. First, Section 10.2 states the datasets that the researcher used for testing and evaluation. Second, Section 10.3 explains the process of prototype development and lists the selected tools for the implementation. Section 10.4 provides an overview of the evaluation process. Sections 10.5 to 10.9 present a detailed test evaluation of Accessibility, Portability, Extensibility, Interoperability and Result Optimisation respectively. Section 10.10 describes the overall evaluation of the proposed system.

10.2 The Test Data

The Comprehensive Knowledge Archive Network (CKAN)[55] is a catalogue of 9,802 structured datasets. Datasets from CKAN are available either in serialised files (e.g. RDF or some other formats) or/and via SPARQL endpoints [Ermilov *et al.*, 2013]. At the time of

[55]http://datahub.io/ - Datahub is a free, powerful data management platform from the Open Knowledge Foundation based on the CKAN data management system.

writing, CKAN provides a list of 103 datasets that provide data in RDF format as well as a SPARQL endpoint. After careful analysis of available datasets, the author found that many of the dataset links were invalid or their SPARQL endpoints were no longer available. The author short listed 12 datasets that provide their RDF dataset, ontology and SPARQL endpoint with an open source license. From further investigation of the 12 available datasets, the author has chosen two datasets as given below.

10.2.1 British Library

The British Library[56] provides British National Bibliography (BNB) resources[57] in RDF format under a Creative Common Universal Public Domain License[58]. The author has chosen British Library datasets because of the data quality, available sample queries and flexible usage limit. The author performed initial tests using the British Library SPARQL endpoint. The sample queries helped the author to verify initial results retrieved by SIRF. British Library uses SPARQL 1.0 and that is why it cannot answer the queries supported by the newer version of SPARQL. However it can still answer a broad range of queries.

10.2.2 DBPedia

The author expanded test data to DBPedia datasets to validate the extensibility and portability of the proposed system. The DBpedia extracts information from Wikipedia and makes it widely available for Linked Data best practices and it is one of the finest samples of collaboratively collected content [Lehmann *et al.*, 2014]. DBPedia is a repository of 4.22 million things [59], including Persons, Organisations, Places, Work and Species. The DBPedia endpoint uses SPARQL 1.1. DBPedia has been used, as a test base, by the majority of the selected semantic search systems (i.e. QAKiS [Cabrio *et al.*, 2012] and PowerAqua [Lopez *et al.*, 2011]).

10.3 Construction of a Prototype

In order to build a prototype for the proposed framework, the author has chosen some existing tools (wherever they are available) to fit in the process. The prototype construction

[56]http://www.bl.uk - British Library.

[57]http://www.bl.uk/bibliographic/datafree.html - British Library Datasets.

[58]http://creativecommons.org/publicdomain/zero/1.0/- Creative Common Universal Public Domain License.

[59]http://wiki.dbpedia.org/about/about-dbpedia/facts-figures - Facts and Figures for DBPedia dataset [accessed January 2016].

required the development of six major modules of SIRF: (i) Data Parser, (ii) Ontology Processor, (iii) Query Optimiser, (iv) Query Formatter, (v) Result Optimiser and (vi) User Interface.

10.3.1 Implementation of the Data Parser

The Data Parser currently deals with RDF data only (as described in Chapter 6 Section 6.3). This module has been written using Java language. The author has chosen Java to build the framework's Knowledge Base for two reasons: (i) Java is capable of reading large files (e.g. files greater than 2MB which is a normal lower limit of RDF dataset files) and (ii) there are many related Java APIs available e.g. RDF API, OWL API and WordNET etc. Also many leading NLP components are already implemented in Java [Finlayson, 2014].

The RDF Parser (in the Data Parser) has been written using Apache Jena RDF API[60]. [Carroll *et al.*, 2004] introduced Jena as an integrated solution for the implementation of the Semantic Web recommendations i.e. RDF and OWL. The RDF Parser parses RDF files and saves instances and Named Entities in the 'Bag of Keywords' (as defined in Chapter 6 Section 6.3). The author has used a MySQL[61] database as a cache repository. MySQL is an extremely well supported open source relational database which can be readily interfaced with languages such as Java. The complete implementation code for the Data Parser can be found in Appendix I (Listing 10). The Data Parser utilises external libraries to map terms to their plural form and synonyms. To map nouns to their plural forms, the author has used Java Inflector library[62] from JBoss DNA[63]. Likewise, for mapping synonyms, the author has utilised Java API for WordNet Searching (JAWS)[64].

10.3.2 Implementation of the Ontology Processor

The Ontology Processor utilises Apache Jena Ontology API[65] to parse ontologies for different domains. This module has also been written in Java. The code for the implementation of this module can be found in Appendix I (Listing 8). The parsed ontology concepts

[60]http://jena.apache.org/documentation/rdf/index.html - A free and open source Java framework for building Semantic Web and Linked Data applications.

[61]https://www.mysql.com/ - MySQL is an open-source relational database management system.

[62]http://modeshape.jboss.org/downloads/downloads-jboss-dna - JBOSS DNA.

[63]JBoss DNA is available under GNU Lesser General Public License (details can be found at http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html)

[64]http://lyle.smu.edu/ tspell/jaws/index.html - Java API for WordNet Searching.

[65]http://jena.apache.org/documentation/ontology/- Apache Jena Ontology API is a tool that parses OWL ontologies.

(i.e. classes and properties) are saved in the Ontology Cache (this has been implemented using a MySQL database).

10.3.3 Implementation of the Query Optimiser

The Query Optimiser module works at runtime to process a user query and so has been written using a server side scripting language. The author has chosen PHP language for this purpose since server side scripting using PHP syntax is efficient in response to requests received by a server [Proctor *et al.*, 2014]. The Syntax Analyser (sub module of the Query Optimiser) uses the PHP library for Stanford Parser (PHP-Stanford-NLP[66]) to process natural language queries. The parsed natural language tokens are processed to generate multiple linked pairs (using Algorithm 3 (Syntactic Processing) in Chapter 7). The implementation of Algorithm 3 has been provided in Appendix II (Listings 16 and 25).

10.3.4 Implementation of the Query Formatter and API

The Query Formatter has also been developed using PHP. This module receives tagged syntactic pairs from the Query Optimiser, finds CPI relations, and generates a matching SPARQL query. The implementation of the Query Formatter algorithm can be found in Appendix II (Listing 14, 19 and 20). The Application Program Interface (API) in SIRF uses an open source SPARQL library[67] by Christopher Gutteridge. The API is responsible for connecting to a SPARQL endpoint, running queries and retrieving results (if available).

10.3.5 Implementation of the Result Optimiser

The Result Optimiser module also works at the server level, and so it has been implemented using PHP. This module makes use of result schemas. SIRF saves schemas based on the most used attributes for a class e.g. schema for a class *Person* will be its most used attributes (at least one) e.g. name, family name or given name. Currently, the schema attributes have been generated manually.

[66] https://github.com/agentile/PHP-Stanford-NLP - PHP library to utilise Stanford-NLP parser.
[67] http://graphite.ecs.soton.ac.uk/sparqllib/ - SPARQL library.

10.3.6 Implementation of the User Interface

As described in Chapter 5, the User Interface of SIRF acts in two ways: (i) as an entry point for direct user queries or (ii) as an API to receive user queries from other search systems. The functionality of the user interface as an API could be tested by using any HTTP client. The author has chosen the REST[68] client since it provides a common structure that makes the code reusable [Upadhyaya, 2014]. The testing of the REST client with the proposed system will be explained in Section 10.8.

10.4 Evaluation Overview

The author has evaluated the performance of SIRF using the prototype (as described in Section 10.3). The evaluation has been done based on the criteria defined in Chapter 4 (i.e. Accessibility, Portability, Extensibility, Interoperability and Result Optimisation). The following sections (10.5 to 10.9) detail the evaluation of each criteria in turn.

10.5 Evaluating Accessibility

Accessibility in this context, is the provision of a natural language interface to allow end users and software agents to access semantic data. This section provides evaluation of existing systems (i.e. QAKiS and FREyA) and the proposed framework (SIRF) to access semantic datasets using natural language queries.

Section 10.5.1 presents accessibility evaluation for QAKiS. Section 10.5.2 presents accessibility evaluation for FREyA. Section 10.5.3 presents accessibility evaluation for SIRF. Finally, Section 10.5.4 provides a conclusive discussion for the overall accessibility testing.

10.5.1 Evaluating QAKiS for Accessibility

The author has tested QAKiS for its capacity to support natural language queries. Below is a list of tests executed using QAKiS demo (at http://qakis.org/qakis/index.xhtml).

[68] Representational State Transfer - REST is an architecture that uses simple Hyper Text Transfer Protocol to make calls between machines

10.5.1.1 Test No. 1

User Query:
"search for books"

Result:

Figure 10.1: QAKiS - Accessibility Test 1 (Source: Author, 2015)

The query did not return any result while DBPedia contains books dataset that can be queried using its SPARQL endpoint. The author verified books' collection at DBPedia by running following SPARQL query at DBPedia SPARQL endpoint(http://dbpedia.org/sparql) and retrieved a list of 6773 books.

```
PREFIX ontology: <http://dbpedia.org/ontology/>
select distinct ?book
where {
   ?book rdf:type ontology:Book
}
```

10.5.1.2 Test No. 2

User Query:
"author of Lord of the Rings"

Result:

Figure 10.2: QAKiS - Accessibility Test 2 (Source: Author, 2015)

QAKiS returned a part of the series "Lord of the Rings" as a result for the above query whereas the intended result was the author name for the book. QAKiS was able to only resolves "Lord of the Ring" in the query. The author also observed that the search on QAKiS is case sensitive. If the above query is written in small case "lord of the rings", it fails.

10.5.1.3 Test No. 3

User Query:

"Find books on a subject e.g. crystallography"

Result:

Figure 10.3: QAKiS - Accessibility Test 3 (Source: Author, 2015)

There was no result returned for the above query.

10.5.1.4 Test No. 4

User Query:
"titles by detective writer Ian Rankin"

Result:

Figure 10.4: QAKiS - Accessibility Test 4 (Source: Author, 2015)

QAKiS returned a wrong result for the above query. The tool identified a named entity (i.e. Ian Rankin) and a property (i.e. writer) in the query. However, it was not able to relate both concepts and randomly picked an author (who was a crime writer) and returned as a result.

10.5.1.5 Test No. 5

User Query:
"List 50 authors born in 1945"

Result:

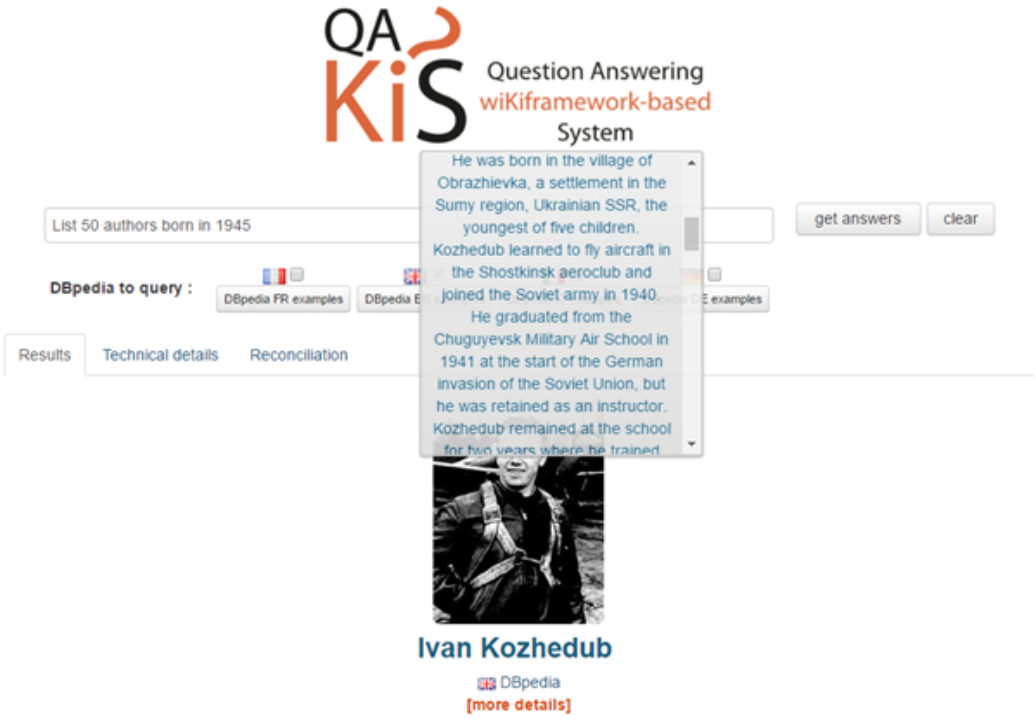


Figure 10.5: QAKiS - Accessibility Test 5 (Source: Author, 2015)

The above query again returned wrong result. The tool identified a named entity (i.e. 1945) and a property (i.e. born) in the query an randomly picked a person and returned as a result (who was neither an author nor born in 1945).

10.5.2 Evaluating FREyA for Accessibility

Below is a list of tests executed to evaluate FREyA's capacity to support natural language queries using its demo (at http://qakis.org/qakis/index.xhtml).

10.5.2.1 Test No. 1

User Query:
"the largest city in California"

Result:

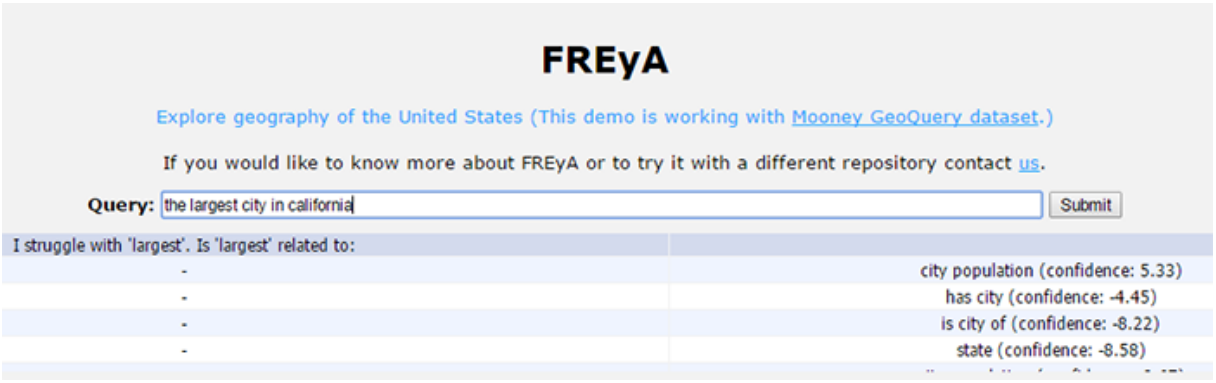


Figure 10.6: FREyA - Accessibility Test 1A (Source: Author, 2015)

Result:

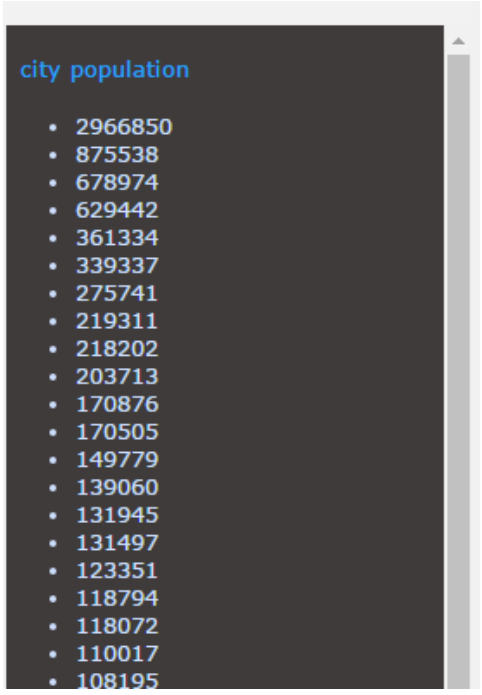


Figure 10.7: FREyA - Accessibility Test 1B (Source: Author, 2015)

FREyA struggled to understand the semantics of the term 'largest' and the relation between 'largest', 'city' and 'California'. On using the clarification dialogue, a list of populations ordered by the highest population on top (Figure 10.7), was retrieved.

10.5.2.2 Test No. 2

User Query:
"find cities and their population"

Result:

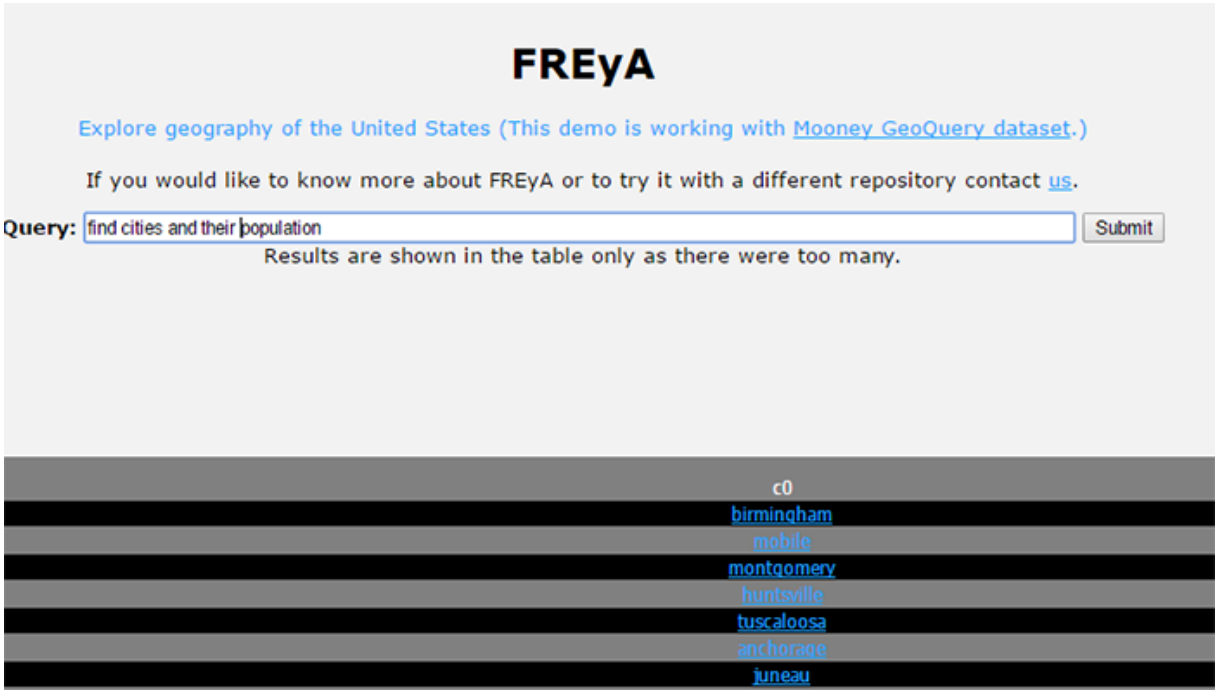


Figure 10.8: FREyA - Accessibility Test 2 (Source: Author, 2015)

For the above query, FREyA was not able to resolve the relation between 'cities' and 'population' and returned a list of cities only.

10.5.3 Evaluating SIRF for Accessibility

The author performed initial tests on British Library (BL) dataset. British Library's developers have provided a set of sample queries for the end users to use their SPARQL endpoint. The author has used these sample queries to validate the effectiveness of SIRF. The author compared the hard coded queries available on BL endpoint with the queries generated by the prototype to evaluate the quality and accuracy of results. The author further experimented with variations of individual queries to validate the accessibility of the data. The details of initial tests are below.

10.5.3.1 Test No. 1

User Query:

"search for a book with ISBN 9780415435864"

Sample SPARQL query from British Library

```
PREFIX bibo: <http://purl.org/ontology/bibo/>
PREFIX bio: <http://purl.org/vocab/bio/0.1/>
PREFIX blt: <http://www.bl.uk/schemas/bibliographic/blterms#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX event: <http://purl.org/NET/c4dm/event.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX isbd: <http://iflastandards.info/ns/isbd/elements/>
PREFIX org: <http://www.w3.org/ns/org#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdau: <http://rdaregistry.info/Elements/u/>
PREFIX madsrdf: <http://www.loc.gov/mads/rdf/v1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX void: <http://rdfs.org/ns/void#>

SELECT ?book ?bnb ?title WHERE {
    ?book bibo:isbn13 "9780415435864";
        blt:bnb ?bnb;
        dct:title ?title.
}
```

Query Results — Visual Results Mode

book	bnb	title
http://bnb.data.bl.uk/id/resource/013799996	GBA757593	The quest for gentility in China : negotiations beyond gender and class

Figure 10.9: Accessibility Test No. 1A - Results from British Library (Source: Author, 2015)

SPARQL Query generated by the Prototype

The author tested following variations of the user query:

Variation 1: "search for a book with ISBN 9780415435864"
Variation 2: "book with ISBN 9780415435864"
Variation 3: "book ISBN 9780415435864"

All three variations generated the same query and results as given below.

```
SELECT DISTINCT * WHERE {
    {?Book  a <http://purl.org/ontology/bibo/Book>}
    {?Book <http://purl.org/ontology/bibo/isbn13> '9780415435864'}
    {?Book <http://purl.org/dc/terms/title> ?Book_title}
}
```

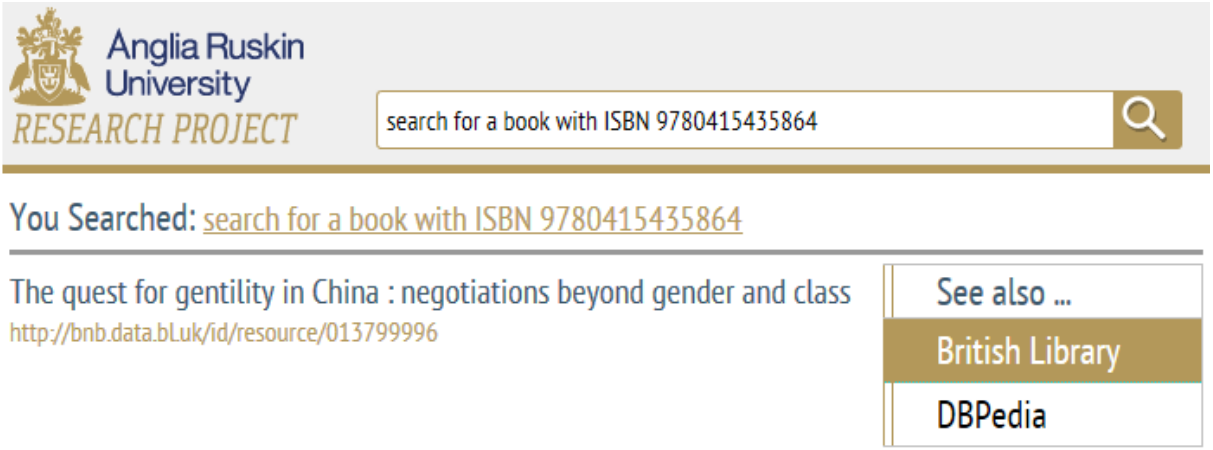


Figure 10.10: Accessibility Test No. 1B - Results from SIRF (Source: Author, 2015)

Discussion:

Figure 10.10 shows that the natural language query tested with the prototype gets similar results to the hard coded query on British Library SPARQL endpoint (Figure 10.9). SIRF extracts the CPI concepts (i.e. Book, ISBN, 9780415435864) and identifies the relation among these concepts. Testing the same query with different text (as described above) returned same result, hence validated the query generation algorithm.

10.5.3.2 Test No. 2

User Query:

"Find serial with ISSN 0955-6664"

Sample SPARQL query from British Library

```
PREFIX bibo: <http://purl.org/ontology/bibo/>
PREFIX bio: <http://purl.org/vocab/bio/0.1/>
PREFIX blt: <http://www.bl.uk/schemas/bibliographic/blterms#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX event: <http://purl.org/NET/c4dm/event.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX isbd: <http://iflastandards.info/ns/isbd/elements/>
PREFIX org: <http://www.w3.org/ns/org#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdau: <http://rdaregistry.info/Elements/u/>
PREFIX madsrdf: <http://www.loc.gov/mads/rdf/v1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX void: <http://rdfs.org/ns/void#>

SELECT ?serial ?bnb ?title WHERE {
   ?serial bibo:issn "0955-6664";
        blt:bnb ?bnb ;
        dct:title ?title.
}
```

Query Results — Visual Results Mode

serial	bnb	title
http://bnb.data.bl.uk/id/resource/007740079	GB9037116	Journal of nutritional medicine

Figure 10.11: Accessibility Test No. 2A - Results from British Library (Source: Author, 2015)

SPARQL Query generated by the Prototype

The author tested following variations of the user query:

Variation 1: "Find book with ISSN 0955-6664"
Variation 2: "book with ISSN 0955-6664"
Variation 3: "book ISSN 0955-6664"

All three variations generated the same query and results as given below.

```
SELECT DISTINCT * WHERE {
    {?Periodical a <http://purl.org/ontology/bibo/Periodical>}
    {?Periodical <http://purl.org/ontology/bibo/issn> '0955-6664'}
    {?Periodical <http://purl.org/dc/terms/title> ?Periodical_title}
}
```

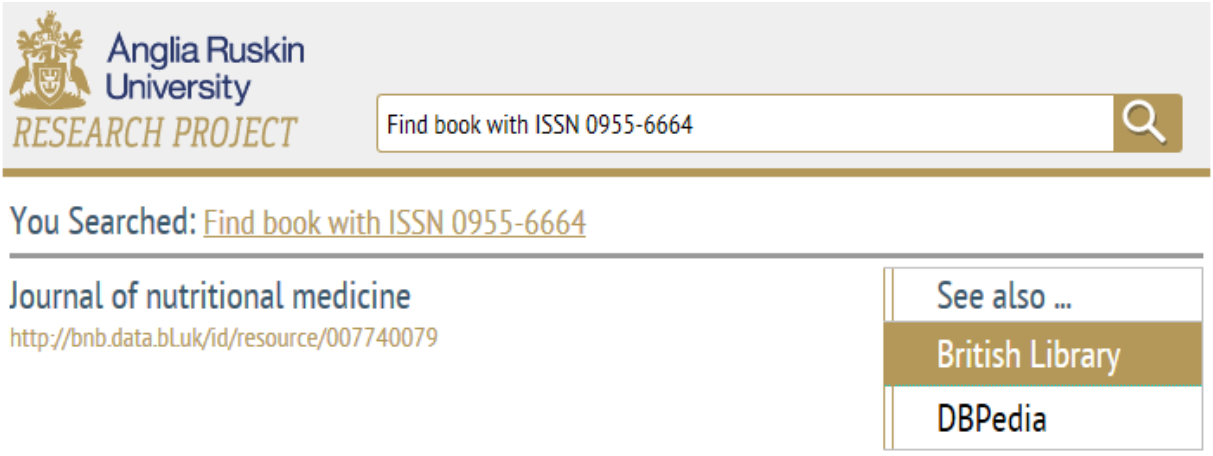


Figure 10.12: Accessibility Test No. 2B - Results from SIRF (Source: Author, 2015)

Discussion:

The above test validated the effectiveness of the prototype to find qualified concepts. The author tested the above query with deliberate non-existing concepts. The query asks for a book with ISSN "0955-6664" whereas an ISSN is linked to periodicals or journals. Figure 10.12 shows that the natural language query tested with the prototype gets similar results to the hard coded query on British Library SPARQL endpoint (Figure 10.11) although the query was only partially matched. The Concept Mapper does not find any relation for "Book" hence ignores it. The Query Generator generates query based on the qualified concepts (i.e. ISSN and "0955-6664").

10.5.3.3 Test No. 3

User Query:

"Find books on a subject e.g. crystallography"

Sample SPARQL query from British Library

```
PREFIX bibo: <http://purl.org/ontology/bibo/>
PREFIX bio: <http://purl.org/vocab/bio/0.1/>
PREFIX blt: <http://www.bl.uk/schemas/bibliographic/blterms#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX event: <http://purl.org/NET/c4dm/event.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX isbd: <http://iflastandards.info/ns/isbd/elements/>
PREFIX org: <http://www.w3.org/ns/org#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdau: <http://rdaregistry.info/Elements/u/>
PREFIX madsrdf: <http://www.loc.gov/mads/rdf/v1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX void: <http://rdfs.org/ns/void#>

SELECT ?book ?isbn ?title WHERE {
   ?book dct:subject <http://bnb.data.bl.uk/id/concept/lcsh/
      Crystallography>;
        bibo:isbn13 ?isbn;
        dct:title ?title.
}
```

Query Results — Visual Results Mode

book	isbn	title
http://bnb.data.bl.uk/id/resource/015351099	9780470699614	NMR crystallography
http://bnb.data.bl.uk/id/resource/016196249	9781439887622	Structure and properties of fat crystal networks
http://bnb.data.bl.uk/id/resource/017050047	9781634637916	Crystals and crystal growth
http://bnb.data.bl.uk/id/resource/017164005	9780198738688	The Basics of Crystallography and Diffraction

Figure 10.13: Accessibility Test No. 3A - Portion of results from British Library (Source: Author, 2015)

SPARQL Query generated by the Prototype System

The author tested following variations of the user query:

Variation 1: “Find books on Crystallography”
Variation 2: “books on Crystallography”
Variation 3: “Crystallography”

All three variations generated the same query and results as given below.

```
SELECT DISTINCT * WHERE {
    {?Book  a <http://purl.org/ontology/bibo/Book>}
   {?Subject <http://www.w3.org/2000/01/rdf-schema#label> "
      Crystallography"}
    {?Book <http://purl.org/dc/terms/subject> ?Subject}
    {?Book <http://purl.org/dc/terms/title> ?Book_title}
}
```

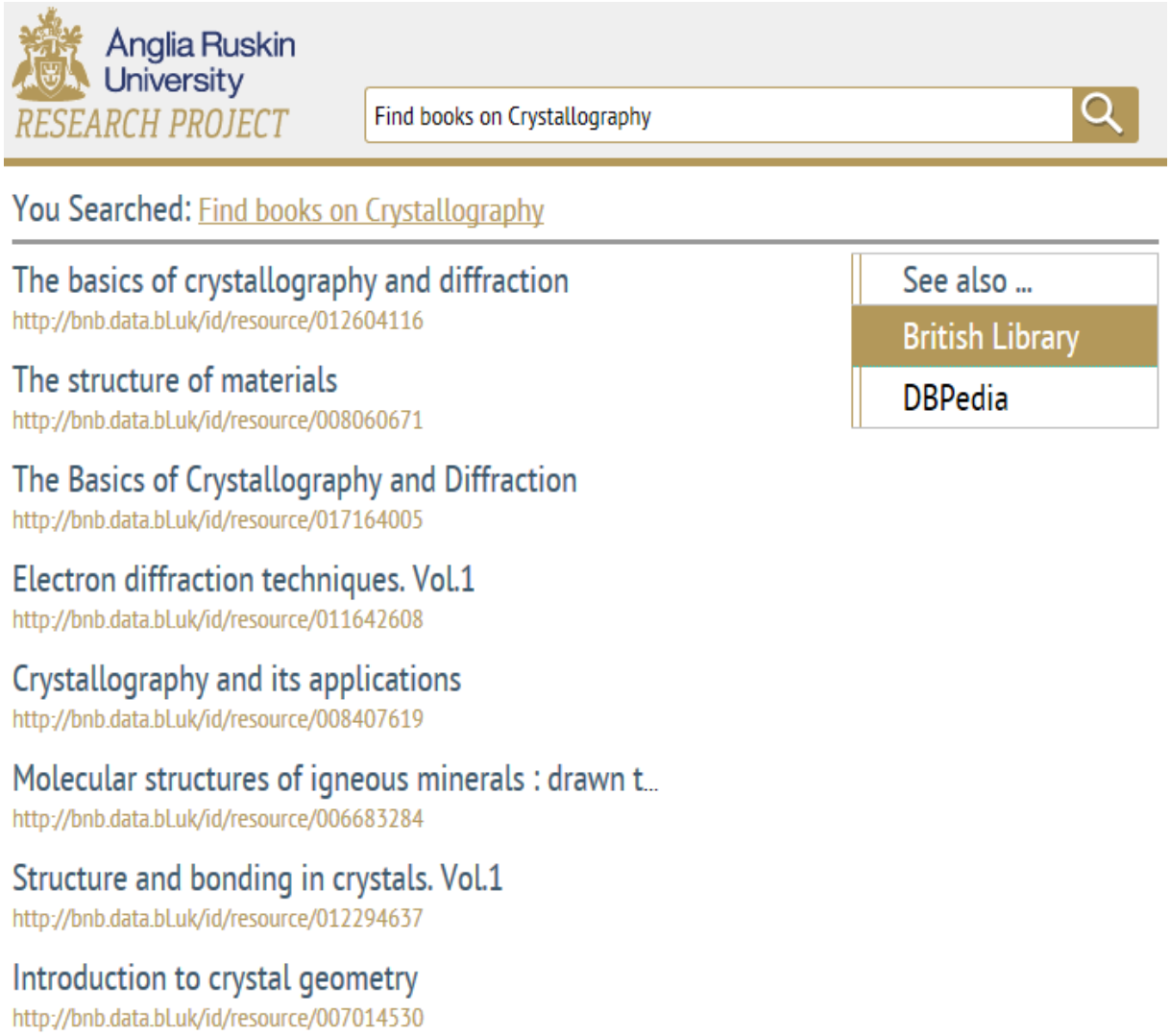


Figure 10.14: Accessibility Test No. 3B - Results from SIRF (Source: Author, 2015)

Discussion:

The above query tests (Figure 10.14) the capacity of the CPI Binder to find missing concepts. The Concept Mapper finds CPI available concepts (i.e. book, crystallography) and calculates the missing concept (i.e. subject) that links "Book" with "crystallography".

10.5.3.4 Test No. 4

User Query:

"titles by detective writer Ian Rankin"

Sample SPARQL query from British Library

```
PREFIX bibo: <http://purl.org/ontology/bibo/>
```

```
PREFIX bio: <http://purl.org/vocab/bio/0.1/>
PREFIX blt: <http://www.bl.uk/schemas/bibliographic/blterms#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX event: <http://purl.org/NET/c4dm/event.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX isbd: <http://iflastandards.info/ns/isbd/elements/>
PREFIX org: <http://www.w3.org/ns/org#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdau: <http://rdaregistry.info/Elements/u/>
PREFIX madsrdf: <http://www.loc.gov/mads/rdf/v1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX void: <http://rdfs.org/ns/void#>

SELECT DISTINCT ?title WHERE {
   ?book dct:creator <http://bnb.data.bl.uk/id/person/RankinIan>;
      dct:title ?title;
}
```

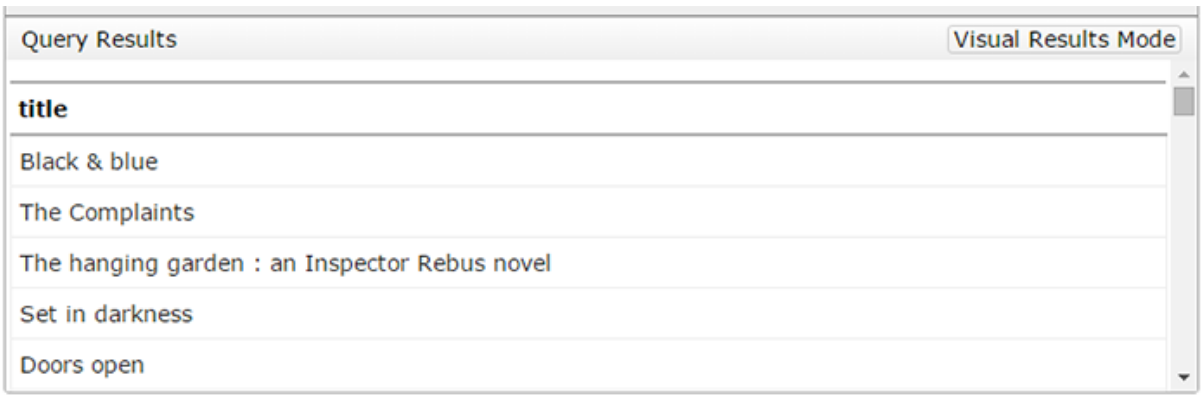


Figure 10.15: Accessibility Test No. 4A - Portion of results from British Library (Source: Author, 2015)

SPARQL Query generated by the Prototype

```
SELECT DISTINCT * WHERE {
    {?Book <http://purl.org/dc/terms/creator> ?Person}
    {?Person a <http://xmlns.com/foaf/0.1/Person>}
    {?Person <http://xmlns.com/foaf/0.1/name> "Ian Rankin"}
    {?Book <http://purl.org/dc/terms/title> ?Book_title}
}
```

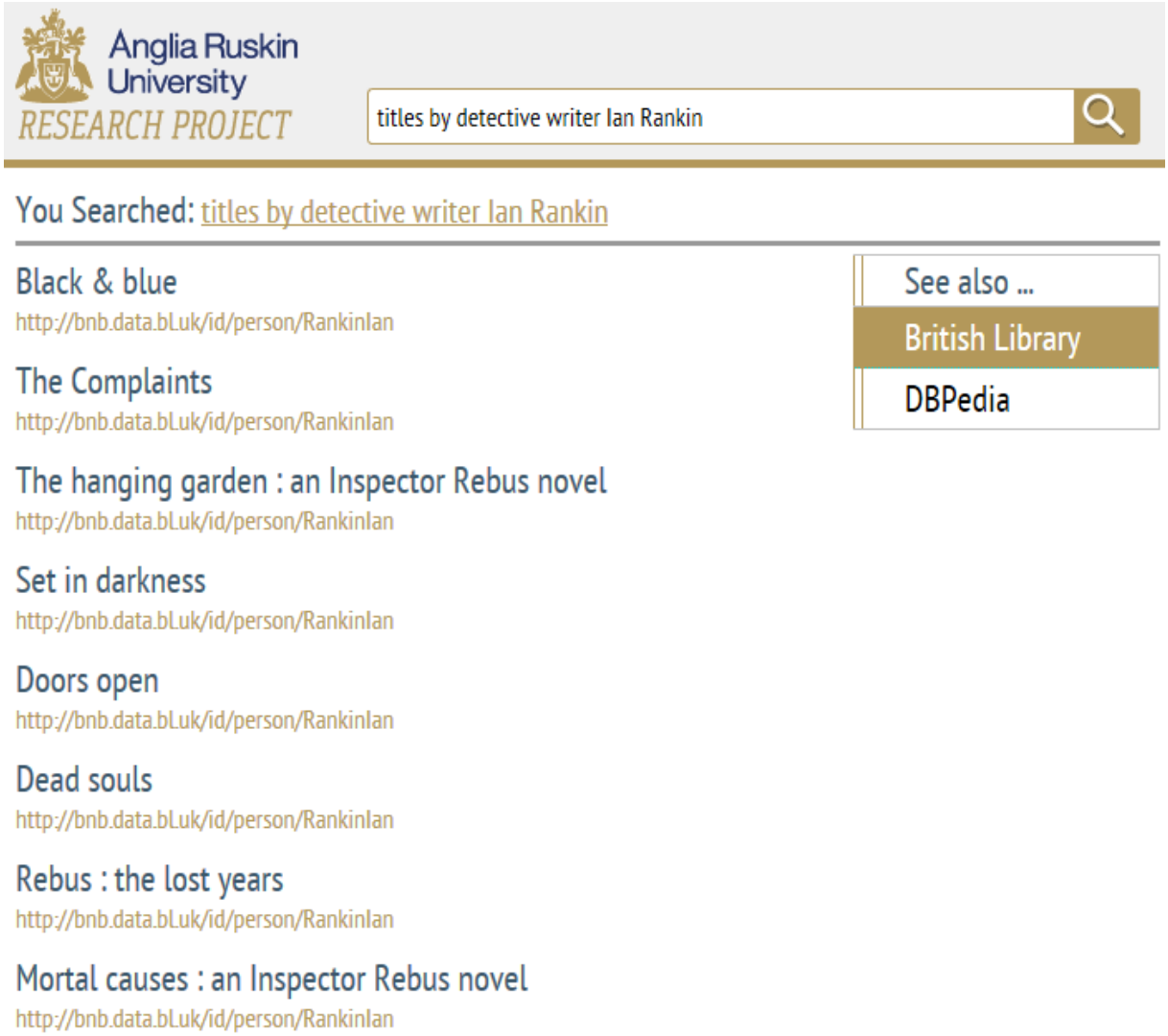


Figure 10.16: Accessibility Test No. 4B - Results from SIRF (Source: Author, 2015)

Discussion:

Figure 10.16 shows the effectiveness of the prototype to identify qualified concepts and ignore the unidentified concepts. The natural language query tested with the prototype gets similar results to the hard coded query on British Library SPARQL endpoint (Figure 10.15). SIRF's concept mapper maps two concepts in the natural language query (i.e. Ian Rankin, writer) and ignores the unidentified ones. Although the above results prove the capacity of the Query Generator to generate a query when some of the concepts are unidentified, the results do not fulfil the intent of the query. From the above results, the author concludes that the semantic search can be limited if concepts are not matched to ontology concepts.

10.5.3.5 Test No. 5

User Query:

"List 50 authors born in 1945"

Sample SPARQL query from British Library

```
PREFIX bibo: <http://purl.org/ontology/bibo/>
PREFIX bio: <http://purl.org/vocab/bio/0.1/>
PREFIX blt: <http://www.bl.uk/schemas/bibliographic/blterms#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX event: <http://purl.org/NET/c4dm/event.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX isbd: <http://iflastandards.info/ns/isbd/elements/>
PREFIX org: <http://www.w3.org/ns/org#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdau: <http://rdaregistry.info/Elements/u/>
PREFIX madsrdf: <http://www.loc.gov/mads/rdf/v1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX void: <http://rdfs.org/ns/void#>

SELECT ?author ?name WHERE {
    ?event a bio:Birth;
        bio:date "1945"^^<http://www.w3.org/2001/XMLSchema#gYear>.
    ?author bio:event ?event;
        foaf:name ?name.
}
LIMIT 50
```

Query Results — Visual Results Mode

author	name
http://bnb.data.bl.uk/id/person/SmithGordon1945-	Gordon Smith
http://bnb.data.bl.uk/id/person/BakerPaul1945-	Paul Baker
http://bnb.data.bl.uk/id/person/GravesRichardPerceval1945-	Richard Perceval Graves
http://bnb.data.bl.uk/id/person/Lombn%C3%A6sAndreasG1945-	Andreas G. Lombnæs
http://bnb.data.bl.uk/id/person/MacdonaldBarrie1945-	Barrie Macdonald

Figure 10.17: Accessibility Test No. 5A - Results from British Library (Source: Author, 2015)

SPARQL Query generated by the Prototype

The author tested following variations of the user query:

Variation 1: “List 50 authors born in 1945”
Variation 2: “Find 50 authors born in 1945”
Variation 3: “Search 50 authors born in 1945”

All three variations generated the same query and results as given below.

```
SELECT DISTINCT * WHERE {
    {?var1 <http://purl.org/dc/terms/creator> ?Person}
    {?Person <http://purl.org/vocab/bio/0.1/event> ?Event}
    {?Event a <http://purl.org/vocab/bio/0.1/Birth>}
    {?Event <http://purl.org/vocab/bio/0.1/date> ?Event_date}
    {?Person <http://xmlns.com/foaf/0.1/name> ?Person_name}
FILTER (?Event_date = "1945"^^<http://www.w3.org/2001/XMLSchema#gYear>)
}LIMIT 50
```

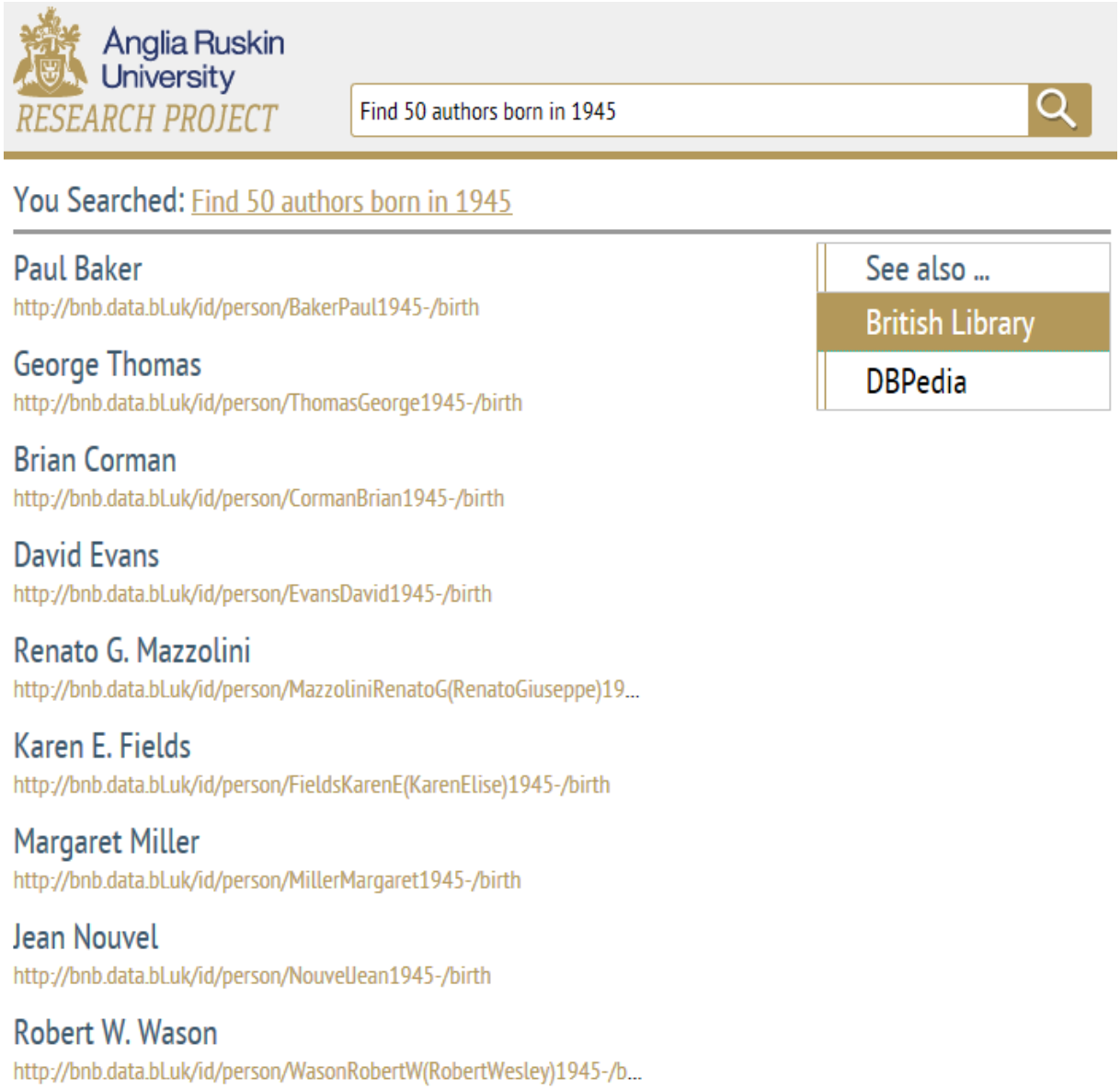


Figure 10.18: Accessibility Test No. 5B - Results from SIRF (Source: Author, 2015)

Discussion:

Figure 10.18 shows the effectiveness of the prototype to identify multiple CPI relations and bind these to generate relatively complex queries. The natural language query tested with the prototype gets similar results to the hard coded query on British Library SPARQL endpoint (Figure 10.17).

10.5.4 Accessibility Conclusion

The accessibility evaluation for the existing semantic search systems (QAKiS and FREyA) showed their strong and weak features. Both of the search tools can access semantic data

using natural language queries. However, they had limitations to understand natural language queries. QAKiS only resolved a user query if it finds a named entity in the query hence it returns no results for the majority of the queries. FREyA used clarification dialogues to find the intent of the user query but was not able to relate concepts.

The accessibility tests with SIRF on the British Library dataset gave very positive results with respect to proper concept mapping and query translation. It can be seen from the comparison of results (in Section 10.5.3) that results for prototype queries are similar to the sample queries (from British Library) though the order of results is different for some queries. The author observed that some complex queries could not find results despite the availability of data e.g. the queries using logical AND (&&) operator. Below is an example of such a query generated by the prototype tool for a user query "list persons whose first name is tim and last name is hardy".

```
SELECT DISTINCT * WHERE {
   {?Person a <http://xmlns.com/foaf/0.1/Person>}
   {?Person <http://xmlns.com/foaf/0.1/givenName> ?Person_givenName}
   {?Person <http://xmlns.com/foaf/0.1/familyName> ?Person_familyName}
FILTER(REGEX(str(?Person_giveName),"tim","i") && REGEX(str(?
   Person_familyName),"hardy","i"))
}
```

The above SPARQL query returns an empty result though the generated query is correct in its syntax. The same query returns results when using OR (||) operator in FILTER statement (as shown in Figure 10.19).

Query Results — Visual Results Mode

Person	given_name	family_name
http://bnb.data.bl.uk/id/person/HardyTim%28TimPaulFrank%29	Tim	Hardy
http://bnb.data.bl.uk/id/person/MaddocksTim	Tim	Maddocks
http://bnb.data.bl.uk/id/person/CantTim	Tim	Cant
http://bnb.data.bl.uk/id/person/MillarTim	Tim	Millar
http://bnb.data.bl.uk/id/person/BrowningTim1976-	Tim	Browning

Figure 10.19: Screenshot from British Library (Source: Author, 2015)

The British Library SPARQL test point uses SPARQL 1.0 that also limits the types of queries to be asked [Rakhmawati *et al.*, 2013]. Some of the retrieved results were also found to be redundant even when using DISTINCT keyword.

10.6 Evaluating Portability

Existing semantic search systems (i.e. FREyA and QAKiS) can be categorised as portable systems per their documentation. The author could not verify the effort involved in setting up a new dataset with these search systems since their demos work with a single dataset only.

SIRF can be used with multiple domains. Although it needs to be set up for each new domain, the setup procedure is automated. The prototype requires the path of RDF files as input to set it up for any new dataset. The prototype then parses the RDF files and initialises its Knowledge Base for the new domain.

The initial parsing and caching can be a little expensive in terms of time (to compute indirect relations and align ontology concepts) and space (to save instances and Named Entities) but it is a one time set up and it is beneficial in multiple ways i.e. efficient information retrieval, proper query translation for domain specific queries and concept mapping.

The prototype was first used on British Library and then it was tested again on DBPedia datasets and ontology.

10.6.1 Evaluating SIRF for Portability

The author tested random user queries with their variations on DBPedia datasets to validate the functionality of the prototype on a different domain than the one it was initially built with (i.e. British Library). The details of tests on DBPedia are below.

10.6.1.1 Portability Test No. 1

User Query:
"What is the population of Pakistan?"

SPARQL Query generated by the Prototype

The author tested following variations of the user query:

Variation 1: "What is the population of Pakistan"
Variation 2: "population of Pakistan"
Variation 3: "Pakistan's population"

All three variations generated the same query and results as given below.

```
SELECT DISTINCT * WHERE {
     {?Country a <http://dbpedia.org/ontology/Country>}
     {?Country <http://dbpedia.org/property/populationEstimate> ?
        Country_population}
      {?Country <http://xmlns.com/foaf/0.1/name> ?Country_name}
 FILTER(LANG(?Country_name) = "en" && REGEX(?Country_name, "pakistan
    ","i"))
 }
```

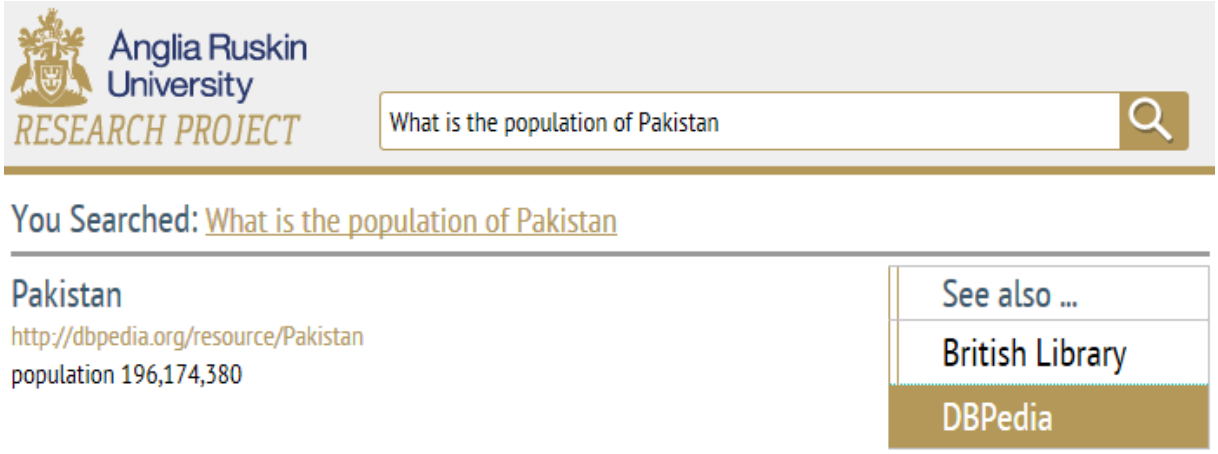


Figure 10.20: SIRF - Portability Test 1 (Source: Author, 2015)

10.6.1.2 Portability Test No. 2

User Query:

"Lord of the Rings"

SPARQL Query generated by the Prototype

```
SELECT DISTINCT * WHERE {
    {?Film rdf:type <http://dbpedia.org/ontology/Film>}
    {?Film <http://www.w3.org/2000/01/rdf-schema#label> ?Film_label}
    FILTER (LANG(?Film_label) = "en" && REGEX(str(?Film_label), "Lord
        of the ring","i"))
 }
```

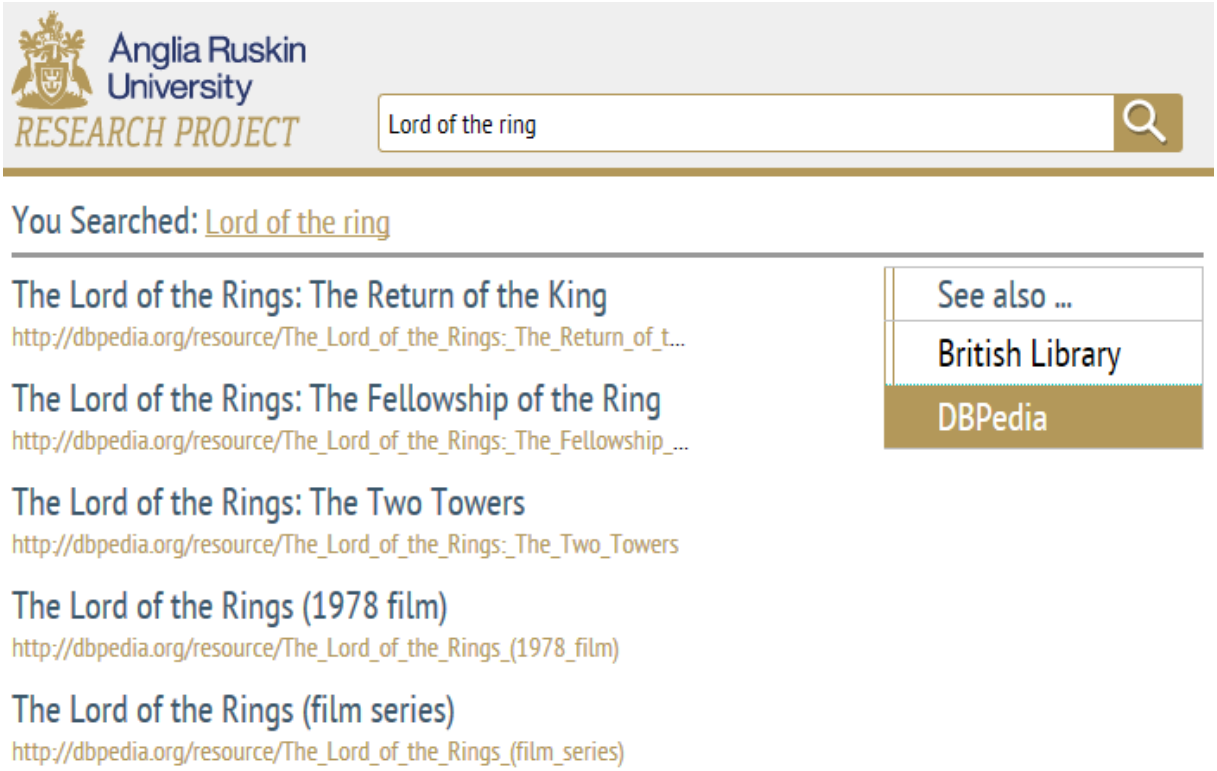


Figure 10.21: SIRF - Portability 2 (Source: Author, 2015)

10.6.2 Portability Conclusion

Setting up SIRF with new domain (i.e. DBPedia) was quite straightforward with no changes required in the code. However, it took a good amount of time (nearly 8 hours) to pre-compute and load data in the database. After the process of parsing DBPedia ontology and RDFs was done, the author executed the above successful tests to verify the capacity of the system to work with multiple datasets.

10.7 Evaluating Extensibility

Existing natural language semantic search tools (e.g. FREyA) currently do not support extensibility. FREyA (Figure 10.22) and QAKiS work with Mooney GeoQuery (Figure 10.22) and DBPedia (Figure 10.23) datasets respectively and can be re-configured to be used with another dataset. However, these are not extensible to merge more than one datasets at once.

Figure 10.22: FREyA Data Source (Source: Author, 2015)

Figure 10.23: QAKiS Data Source (Source: Author, 2015)

To evaluate the extensibility of SIRF i.e. its ability to handle multiple domains simultaneously, the author added multiple domains (i.e. British Library and DBPedia) to the system in a serial fashion. In the first parse, the Data Parser successfully parsed and cached 53,862 unique instances from the British Library BNB dataset. In the second round, the Data Parser parsed 189,700 unique instances from DBPedia. The prototype managed to correctly map the concepts linked with *owl:sameAs* property e.g. dbo:Book and bibo:Book. Figure 10.24 shows the alignment of similar concepts.

class	title	schema	group_id
http://xmlns.com/foaf/0.1/Image	Image	foaf	1
http://purl.org/ontology/bibo/Image	Image	bibo	1
http://semantira.com/Image-foaf	Image	sem	1
http://semantira.com/Project-foaf	Project	sem	2
http://xmlns.com/foaf/0.1/Project	Project	foaf	2
http://xmlns.com/foaf/0.1/Agent	Agent	foaf	3
http://purl.org/dc/terms/Agent	Agent	dct	3
http://semantira.com/Agent-foaf	Agent	sem	3
http://semantira.com/LabelProperty-foaf	LabelProperty	sem	4
http://xmlns.com/foaf/0.1/LabelProperty	LabelProperty	foaf	4
http://purl.org/ontology/bibo/Document	Document	bibo	5
http://semantira.com/Document-foaf	Document	sem	5
http://xmlns.com/foaf/0.1/Document	Document	foaf	5
http://xmlns.com/foaf/0.1/OnlineAccount	OnlineAccount	foaf	6
http://semantira.com/OnlineAccount-foaf	OnlineAccount	sem	6

Figure 10.24: Aligned Concepts (Source: Author, 2015)

10.7.1 Evaluating SIRF for Extensibility

To evaluate the extensibility of the prototype, a number of tests were made for the concepts that exist on both domains e.g. Persons and Books. Following are some examples of tests after merging both domains.

10.7.1.1 Extensibility Test No. 1

User Query:
"List books for Mathematics"

SPARQL query for British Library

```
SELECT DISTINCT *  WHERE {
       {?Book a <http://purl.org/ontology/bibo/Book>}
       {?Book <http://purl.org/dc/terms/title> ?Book_title}
FILTER(REGEX(str(?Book_title), "mathematics","i"))
}
LIMIT 10
```

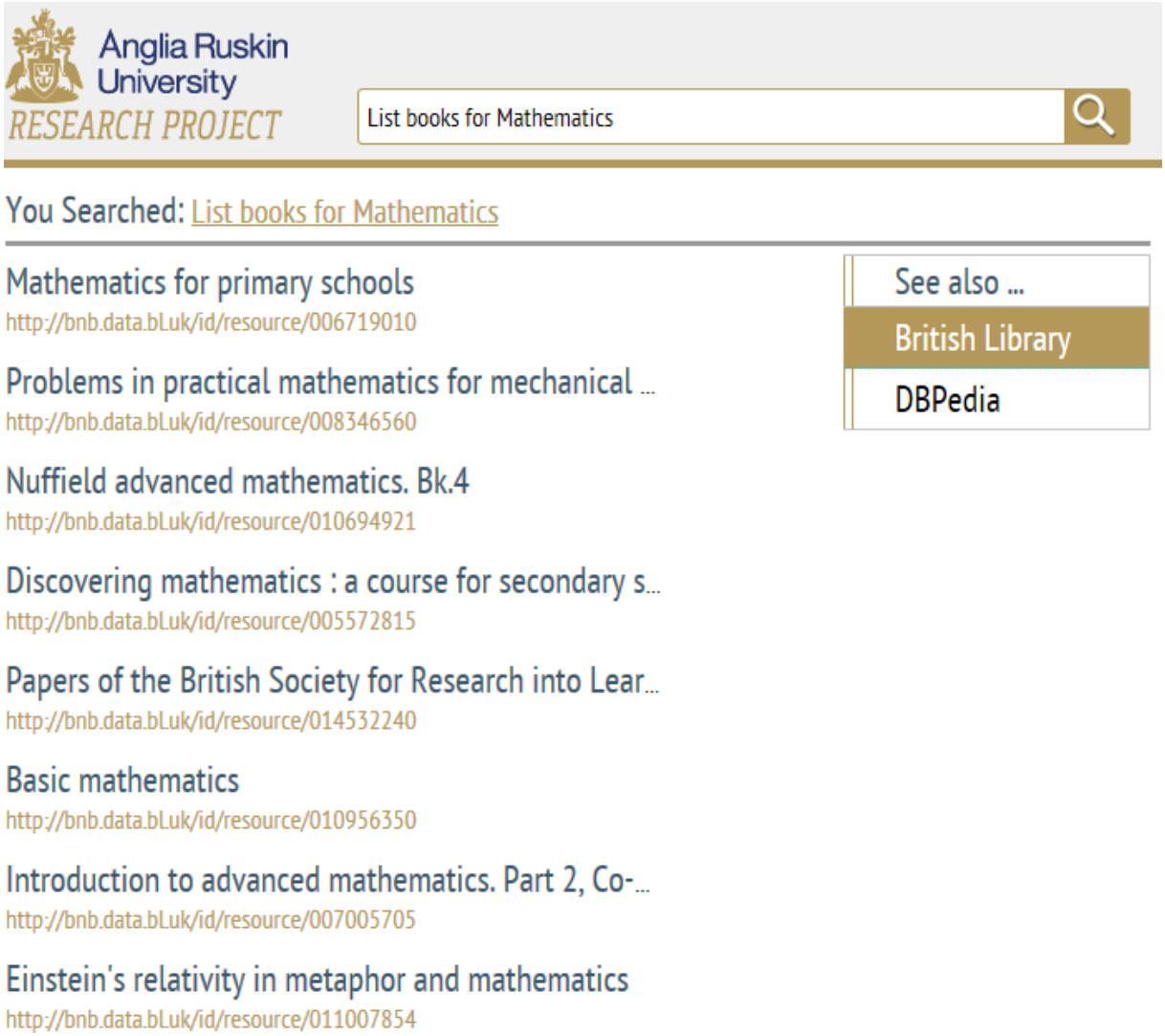


Figure 10.25: Extensibility Test No. 1A - British Library (Source: Author, 2015)

SPARQL query for DBPedia

```
SELECT DISTINCT *  WHERE {
        {?Book a <http://dbpedia.org/ontology/Book>}
        {?Book <http://xmlns.com/foaf/0.1/name> ?Book_name}
        {?Book <http://dbpedia.org/ontology/thumbnail> ?Book_thumbnail}
FILTER(LANG(?Book_name) = "en" && REGEX(str(?Book_name), "mathematics","
    i"))
} LIMIT 10
```

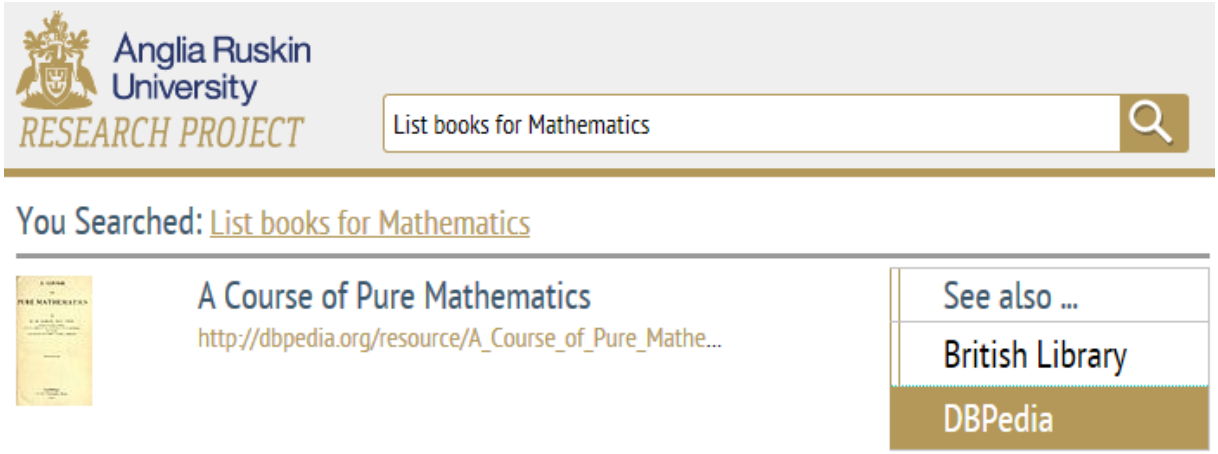


Figure 10.26: Extensibility Test No. 1B - DBPedia (Source: Author, 2015)

10.7.1.2 Extensibility Test No. 2

User Query:
"simon books"

SPARQL query for British Library

```
SELECT DISTINCT *  WHERE {
       {?Book a <http://purl.org/ontology/bibo/Book>}
       {?Person a <http://xmlns.com/foaf/0.1/Person>}
       {?Person <http://xmlns.com/foaf/0.1/familyName> ?
          Person_familyName}
       {?Person <http://xmlns.com/foaf/0.1/givenName> ?Person_givenName}
       {?Book <http://purl.org/dc/terms/creator> ?Person}
       {?Book <http://purl.org/dc/terms/title> ?Book_title}
       {?Person <http://xmlns.com/foaf/0.1/name> ?Person_name}
FILTER((REGEX(str(?Person_familyName), "simon","i") || REGEX(str(?
   Person_givenName), "simon","i")))
}
```

SPARQL query for DBPedia

```
SELECT DISTINCT *  WHERE {
       {?Book a <http://dbpedia.org/ontology/Book>}
       {?Person a <http://xmlns.com/foaf/0.1/Person>}
       {?Person <http://xmlns.com/foaf/0.1/surname> ?Person_surname}
       {?Person <http://xmlns.com/foaf/0.1/givenName> ?Person_givenName}
       {?Book <http://dbpedia.org/ontology/author> ?Person}
       {?Book <http://xmlns.com/foaf/0.1/name> ?Book_name}
       {?Book <http://dbpedia.org/ontology/thumbnail> ?Book_thumbnail}
       {?Person <http://xmlns.com/foaf/0.1/name> ?Person_name}
```

```
FILTER(LANG(?Person_surname) = "en" && (REGEX(str(?Person_surname), "
    simon","i") || REGEX(str(?Person_givenName), "simon","i")))
}
```

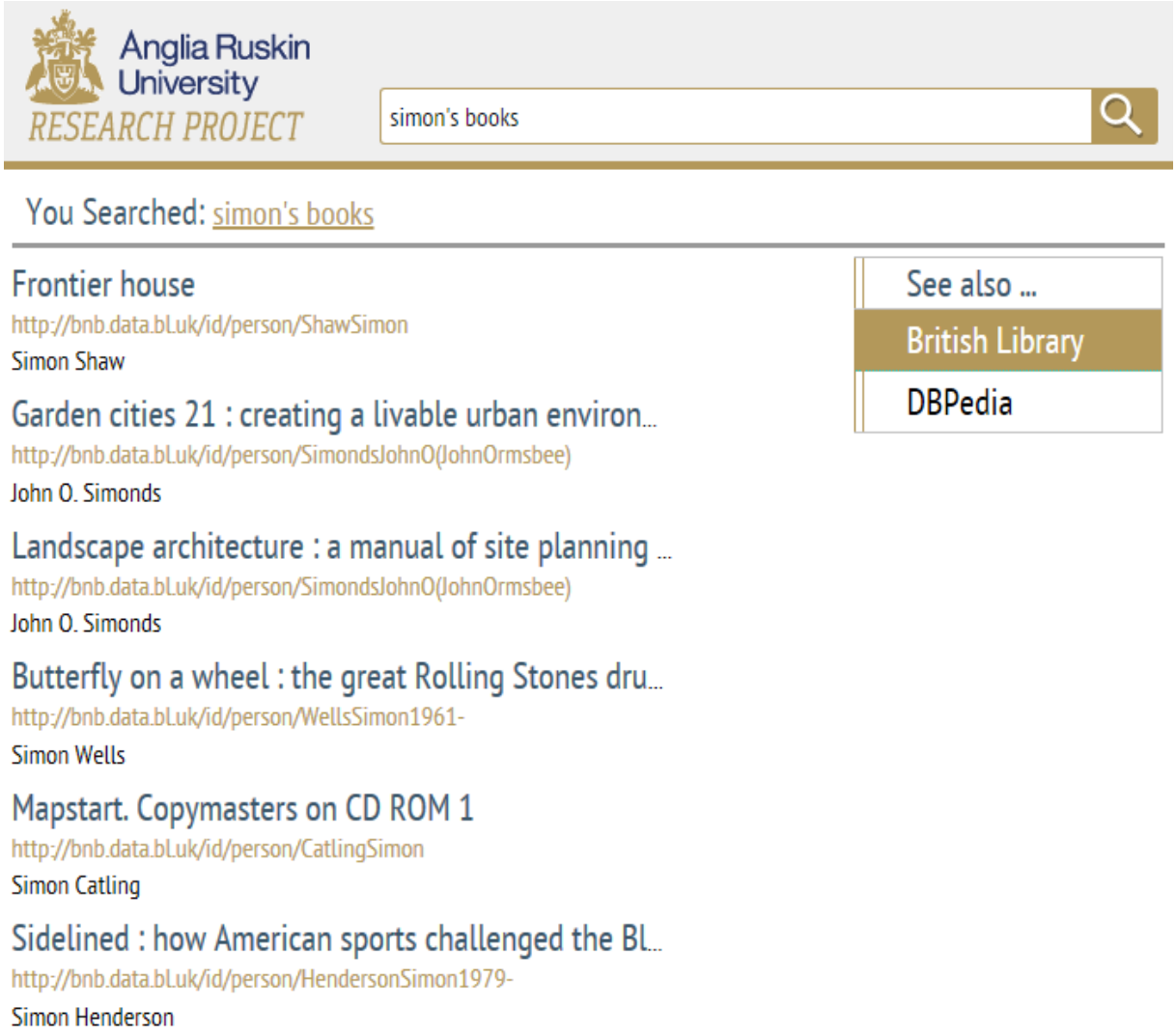


Figure 10.27: Extensibility Test No. 2A - British Library (Source: Author, 2015)

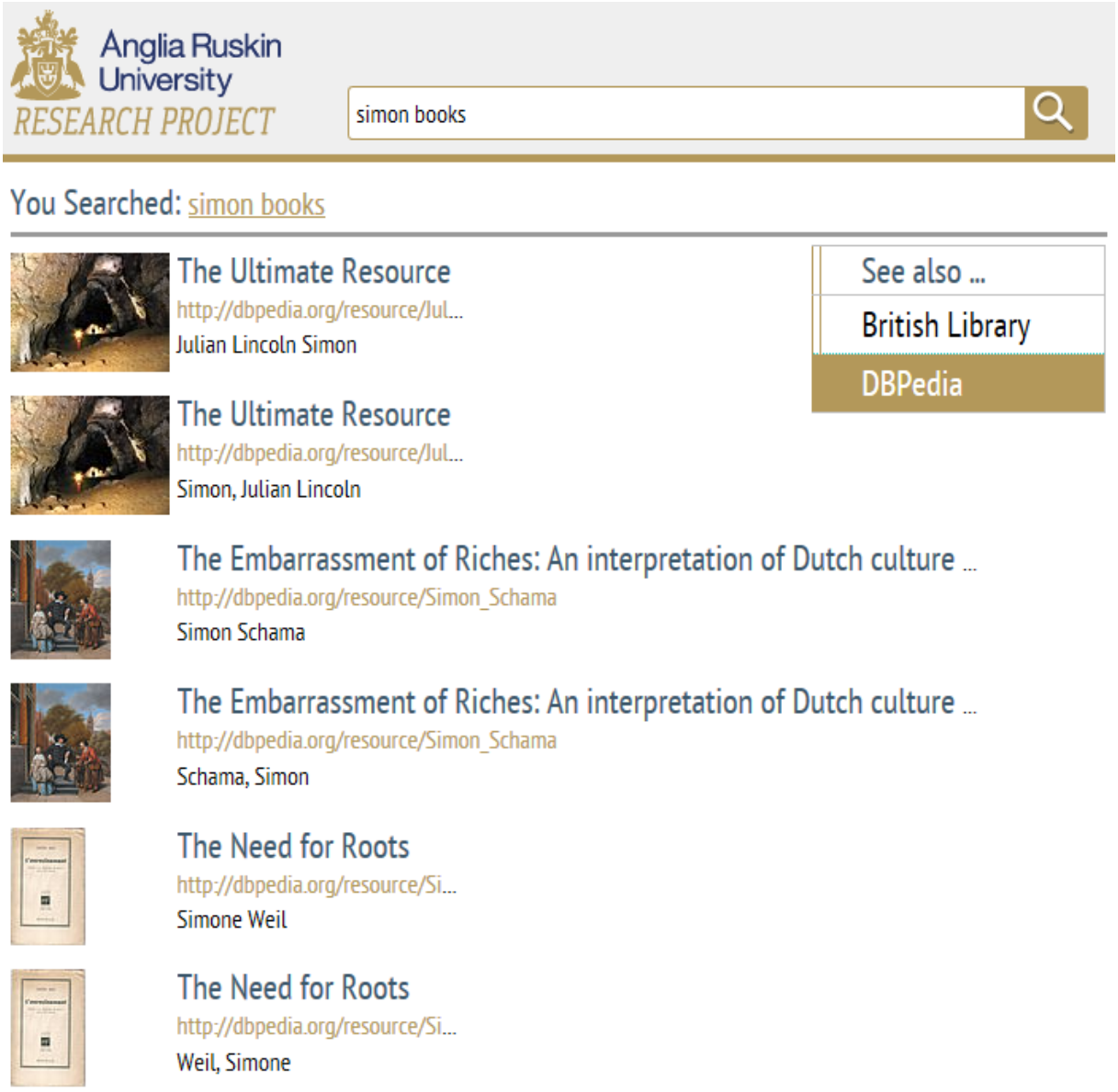


Figure 10.28: Extensibility Test No. 2B - DBPedia (Source: Author, 2015)

10.7.2 Extensibility Conclusion

The author has run a number of queries across both domains (i.e. British Library and DBPedia) using SIRF's prototype. Most of the queries were able to return results but redundant results were found for open constraint queries e.g. "simon's books" where simon can be first name of a person or surname of a person. Figures 10.27 and 10.28 show the results for the above query ("simon's books"). The redundant results are caused by redundant data. Figure 10.28 shows that the books are saved with variations of their authors' names e.g. "The Ultimate Resource" has been listed twice, once with "Julian Lincoln Simon" and then with "Simon, Julian Lincoln". This kind of situations can be

handled in multiple ways e.g. DISTINCT keyword or GROUP BY clause. However it incurs some extra effort and care to clean results. The keyword DISTINCT cannot work when the result set has redundant data in form of data variations. Limiting DISTINCT to a particular resultset variable may lose some valuable data e.g. limiting result set to distinct URIs for books may lose the authors information if there are more than one author.

10.8 Evaluating Interoperability

Application Program Interfaces (APIs) are a common way of increasing the interoperability of various systems. Existing semantic search systems currently do not support interoperability. FREyA uses clarification dialogues that limit its integration with other search tools. To enable the proposed system to interoperate with other search systems, the User Interface of SIRF acts as an API that communicates with the search systems and the LOD datasets. There are a number of ways to design APIs and the simplest is the REST API [Masse, 2011]. The interoperability of SIRF with other search systems, has been tested by using the Advanced REST client for Chrome[69]. Figure 10.29 shows an example of interoperability test where a client system sends a user query "simon books" and the API returns results in JSON format. Thus the prototype has been demonstrated to be capable of interoperability.

[69]https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo - REST Client from Chrome browser.

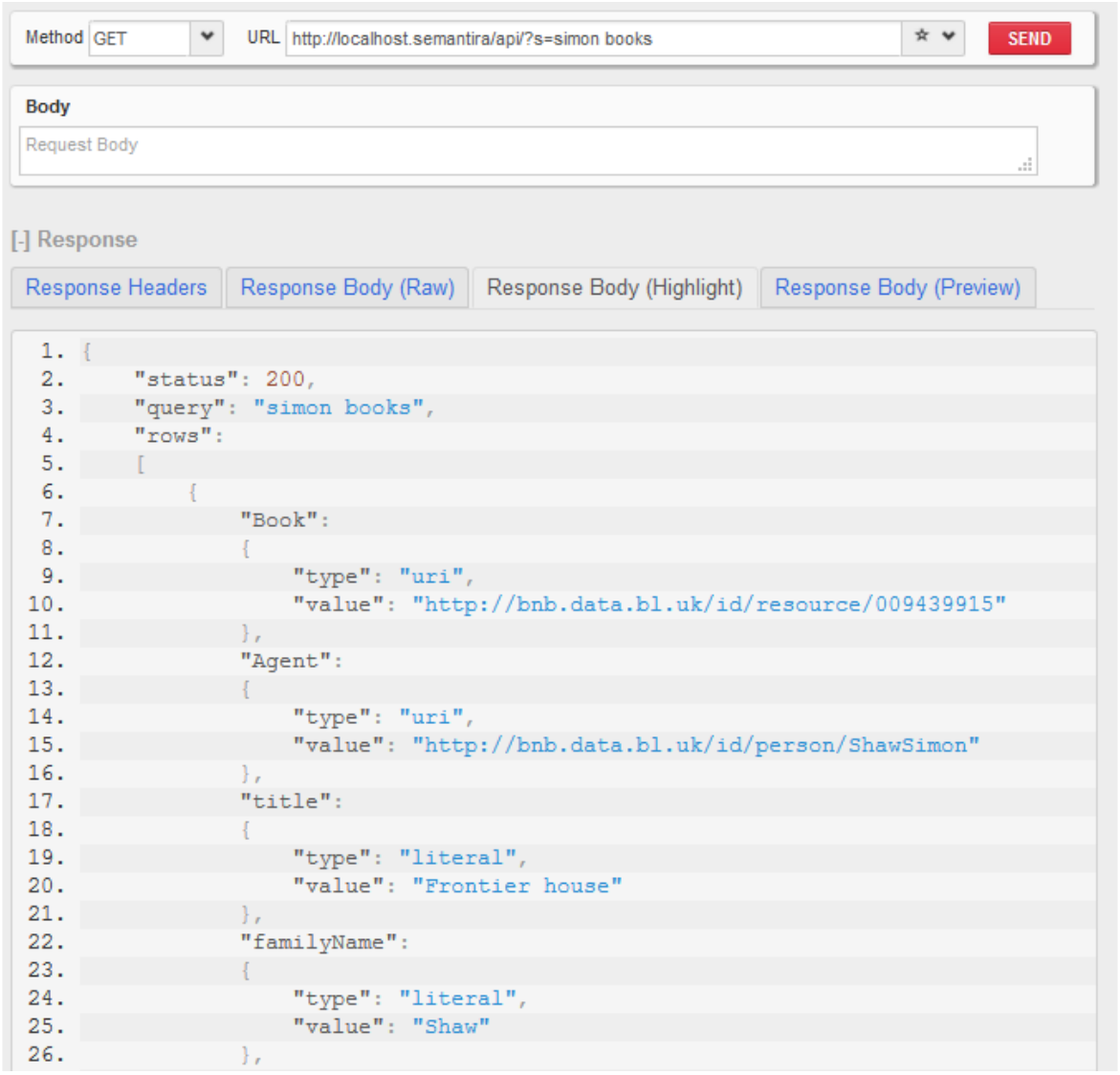


Figure 10.29: Testing Interoperability - REST Client (Source: Author, 2015)

10.9 Evaluating Result Optimisation

Result Optimisation is one of the important features to improve quality of search. Existing natural language semantic search tools are more focused on information retrieval hence ignore this aspect of the search experience. Section 10.9.1 presents evaluation of QAKiS for result optimisation. Section 10.9.2 presents evaluation of FREyA for result optimisation. Section 10.9.3 presents evaluation of SIRF for result optimisation. Finally Section 10.9.4 sums up the conclusions for evaluating result optimisation of the above systems.

10.9.1 Evaluating QAKiS for Result Optimisation

QAKiS seems to deal with result optimisation and presents results in a readable format (Figure 10.30). However, for a majority of queries it brings no results (Figure 10.31). Since the QAKiS framework mainly relies on named entities, it attempts to find and display a picture and a title for that named entity. The results presented by QAKiS are very pleasant aesthetically. Also, having a picture in result gives a quick overview of the result found. In case QAKiS does not find a picture, it displays a standard grey camera image.

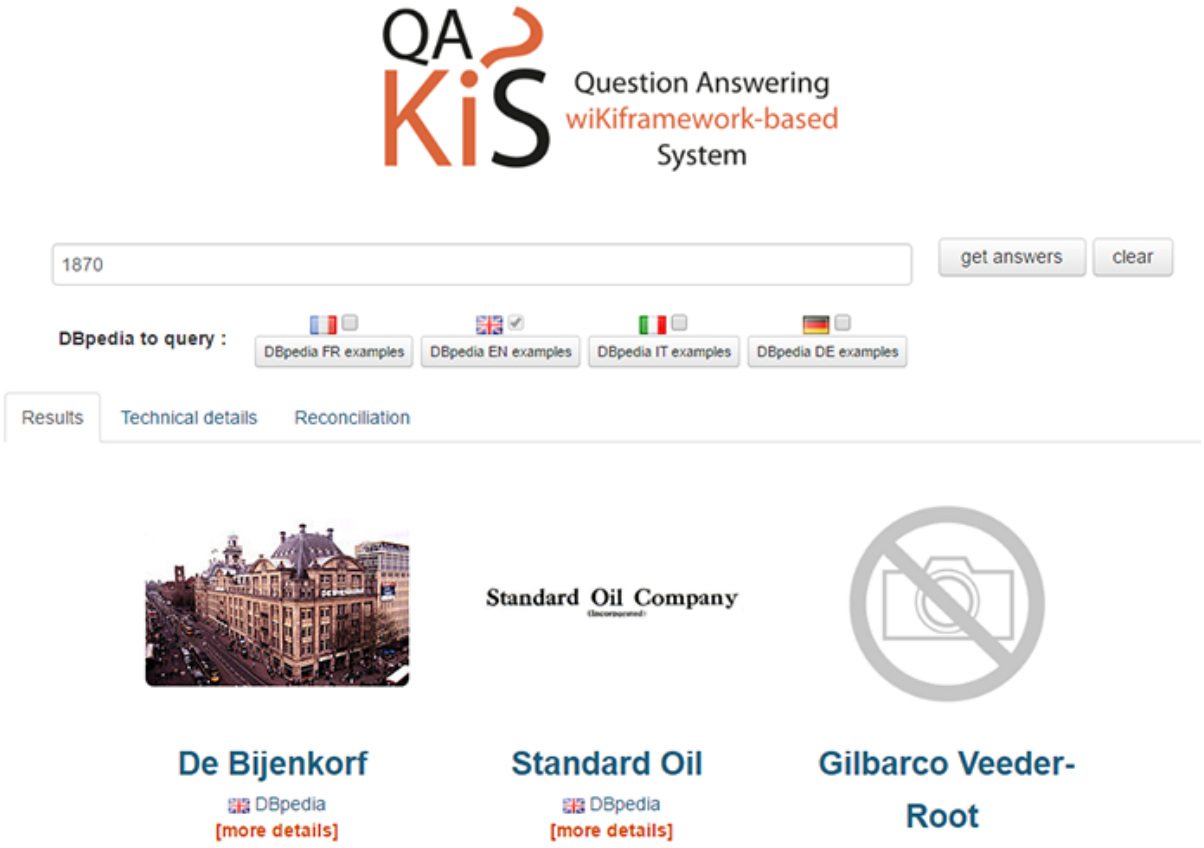


Figure 10.30: QAKiS Result Display 2 (Source: Author, 2015)

Figure 10.31: QAKiS Result Display 1 (Source: Author, 2015)

10.9.2 Evaluating FREyA for Result Optimisation

FREyA displays results in graphical format (Figures 10.32 and 10.33) or a list. FREyA works on Mooney GeoQuery datasets. A graphical presentation like FREyA may work well to represent geographical data. However, it may not be an ideal format to search content rich data.

Figure 10.32: FREyA Result Display (Source: Author, 2015)

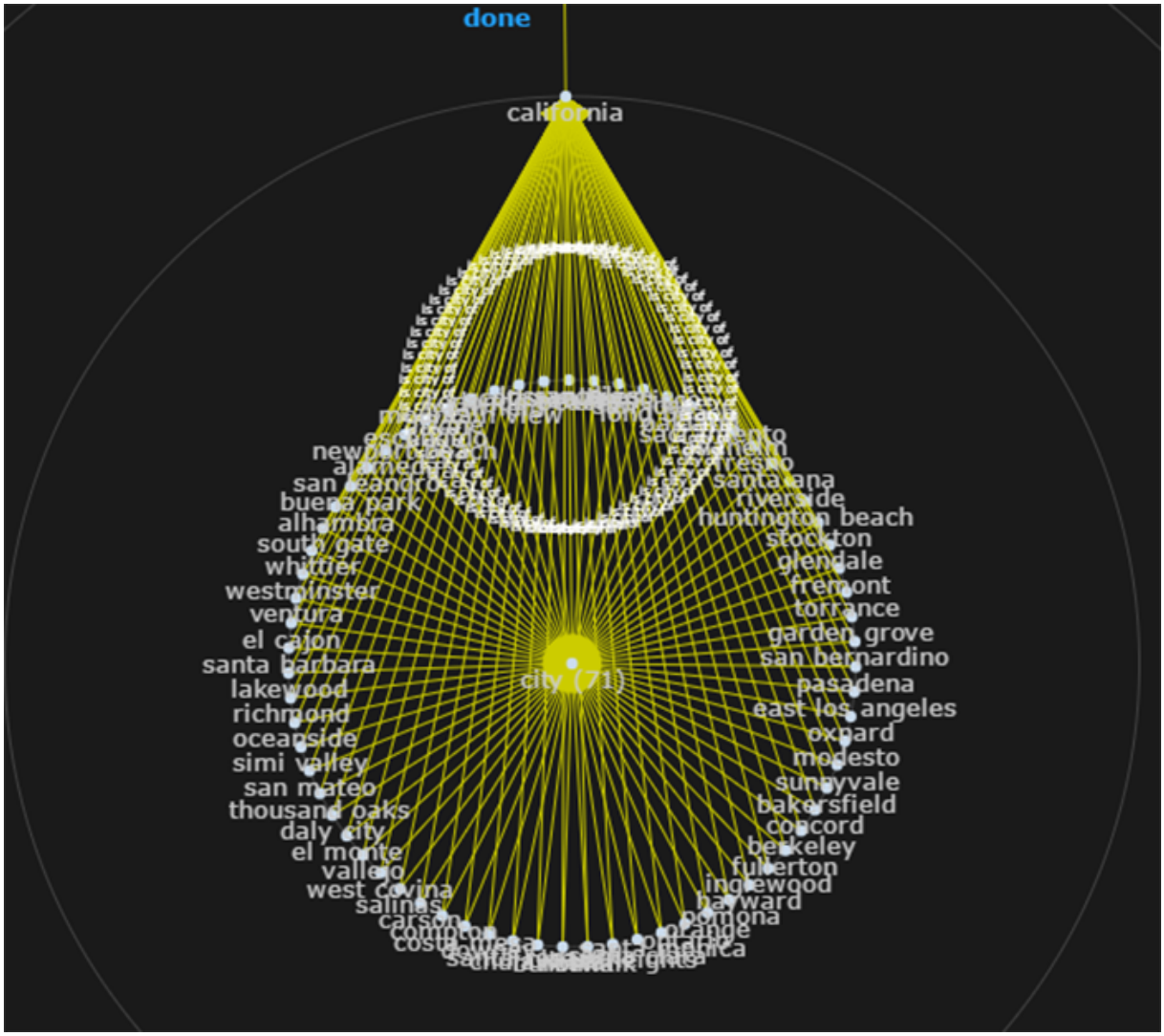


Figure 10.33: FREyA Result Display (Source: Author, 2015)

10.9.3 Evaluating SIRF for Result Optimisation

SIRF handles two aspects of result optimisation i.e. Result Ranking and Result Readability. The process of Result Ranking is handled at the time of query formation to query the domains in order of their ranking. For example for a query to find *Books*, the British Library will be ranked on top since it has higher number of instances (for the concept *Book*) than DBPedia. Readability in this context measures the ease for users to understand the textual controls of a user interface [Ferré, 2014]. The author has introduced pre-defined schemas to display results. A schema defines a set of concepts to be requested from the LOD dataset for a particular CAT (Calculated Answer Type). For example, for a query "List all books for physics" the CAT will be *Book* and the schema attributes will be calculated for CAT *Book*. The Result Optimiser will dispatch the schema attributes

for the concept *Book* to the Query Formatter that will add schema attributes to the SPARQL query (as explained in Chapter 8 and 9). Apart from the basic schemas (as shown in the tests above), the author tried adding some richer schemas to see how it makes a difference to the visualisation and readability. Figure 10.34 shows an example of results from DBPedia after adding richer schema for the concept *Book* i.e. book title, book URI, book thumbnail and book author. Figure 10.35 shows an example of results from British Library after adding richer schema for the concept *Book* i.e. book title, book URI and book author. The prototype will omit the concepts from richer schema that are not found on a domain e.g. in the given example, the prototype will omit thumbnail from the schema for British Library.

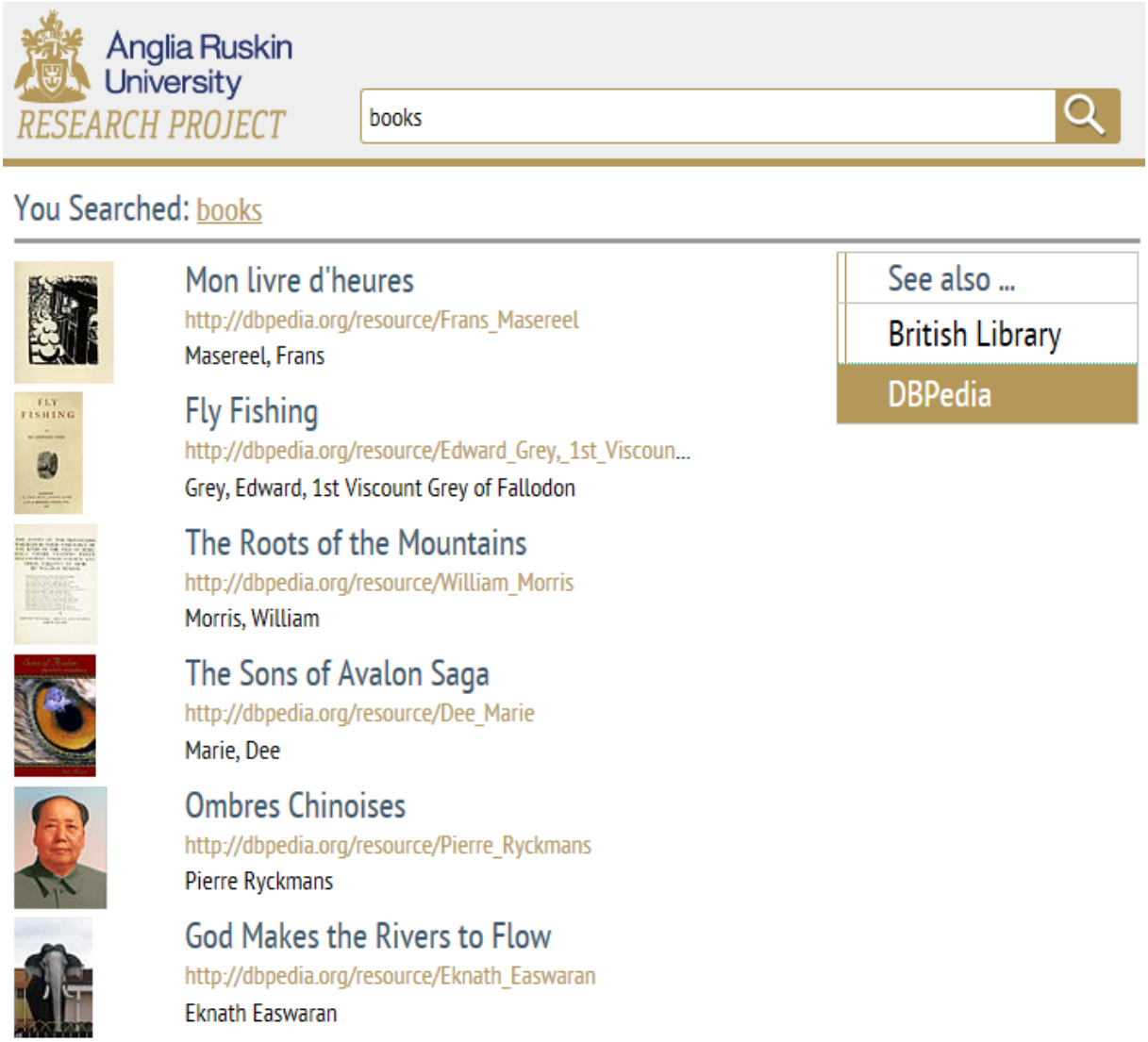


Figure 10.34: Result Optimisation Test No. 1A - DBPedia (Source: Author, 2015)

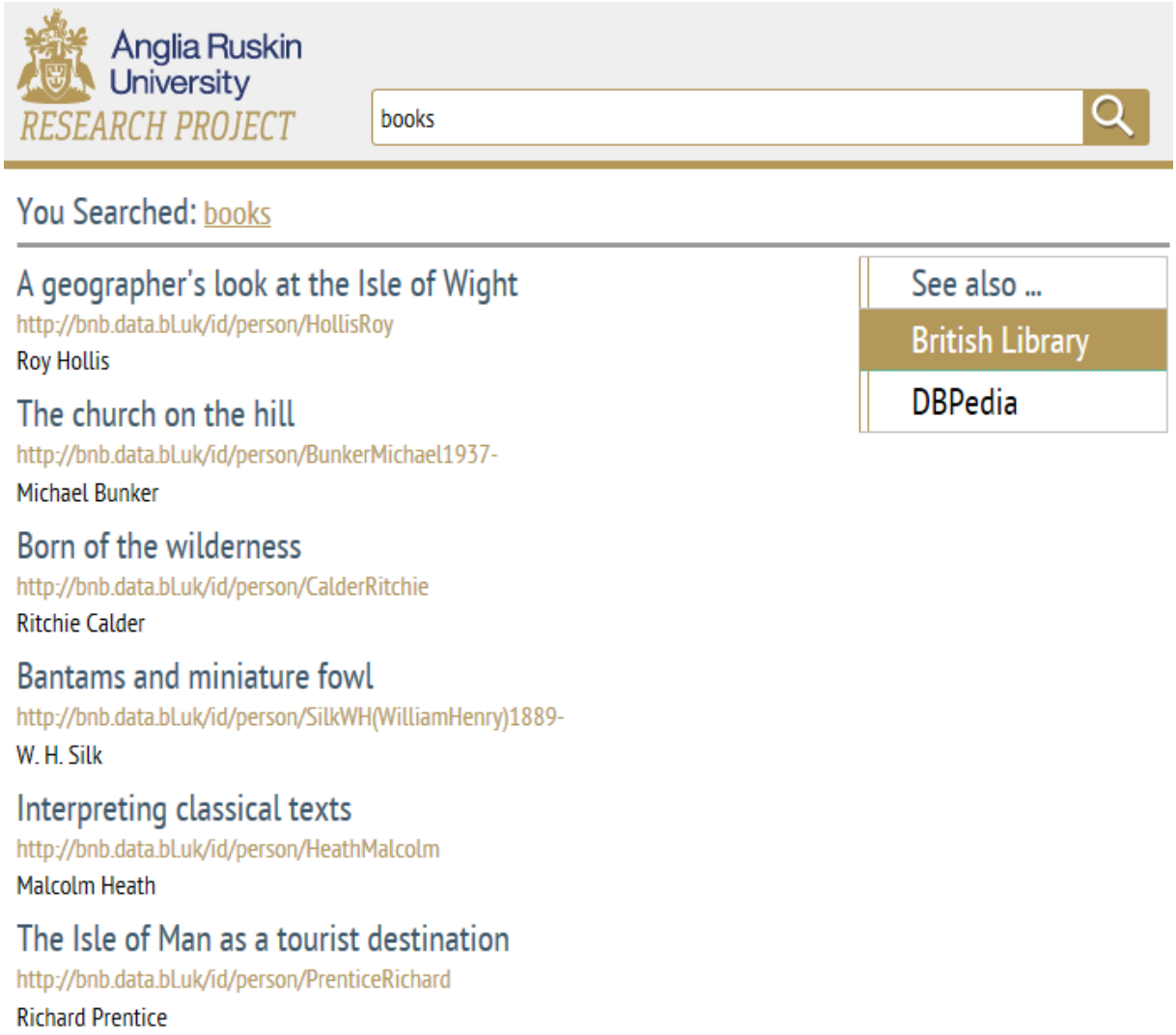


Figure 10.35: Result Optimisation Test No. 1B - British Library (Source: Author, 2015)

10.9.4 Result Optimisation Conclusion

This research deals with two features of result optimisation i.e. result readability and result ranking. Unlike other semantic search systems, QAKiS and FREyA do not present results as a list of URIs. FREyA provides a graphical representations of results. The results format presented by FREyA may not be appropriate for all types of datasets especially text based data. QAKiS presents images and titles of named entities in the result set. This type of representation may fail when expected results are different e.g. for a query 'distance between London and Cambridge', user will expect to see the distance straightaway. For different queries the displayed result may be different e.g. title and image for a query asking for books, name, date of birth, date of death for a query searching for authors etc. To facilitate various queries, SIRF introduced result schemas for resources.

The prototype presents results based on schema e.g. if the CAT (calculated answer type) is a person, the result will display its first name, surname and link. Using schemas for result presentation worked very well for the above tests (Section 10.9.3). However, currently schemas are added manually in the database.

Since QAKiS and FREyA both work with single datasets, these do not require result ranking to display results in a particular order by the dataset. Since SIRF can handle more than one dataset at a time, it caters for both aspects of result optimisation i.e. Result Ranking and Result Readability. It ranks result based on the usage frequency of domain ontologies and the number of instances.

10.10 Overall Evaluation

The prototype Ontology Processor parsed and cached 907 classes (247 from British Library and 660 from DBPedia otology) and 59,817 properties (1072 from British Library and 58,745 from DBPedia). The prototype successfully mapped the majority of the classes and assigned proper groups. However 73 out of 807 classes remained unclassified due to the lack of any links to *owl:sameAs* attributes. That suggests that the concept mapping may fail if *owl:sameAs* is not properly defined.

The author tested 50 random queries (plus their variations) to validate the functionality of the prototype. The evaluation obtained a success rate of about 78%. An illustrative list of queries have been appended in Appendix III.

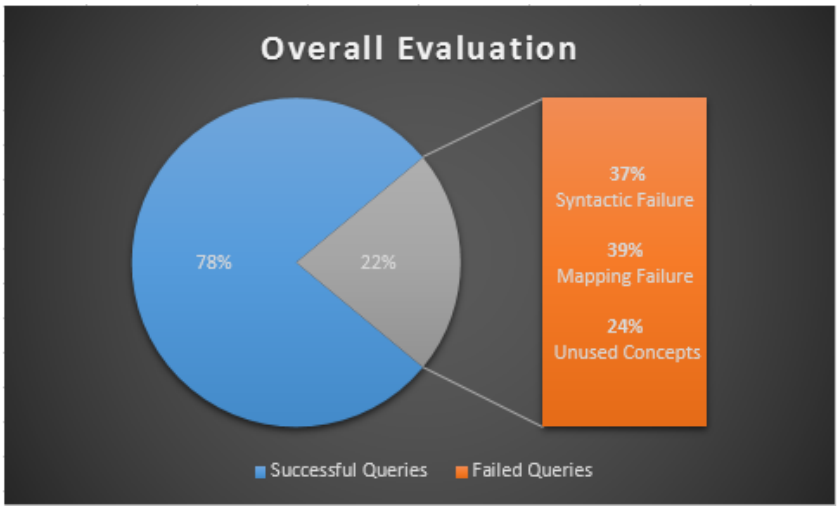


Figure 10.36: Overall Evaluation (Source: Author, 2015)

10.10.1 Evaluation Analysis

The author concluded the following observations from the evaluation:

- SIRF performed well for well structured natural language queries. 78% of tested queries returned expected results. In addition to well structured questions, SIRF was able to resolve simple keywords based user queries (having a maximum of one or two concepts) e.g. "simon's books", "cities in pakistan" etc. However for complex keywords based queries (having multiple concepts), SIRF could not return correct answers due to syntactic failure.
- About 37% of the failed queries were caused by syntactic failure e.g. for a query "largest city pakistan" the NLP parser produced the following dependencies

  (pakistan, city),
  (largest, pakistan)

  The query generated based on the above parse, simply returned list of all cities in Pakistan (since it could not find a relation between 'largest' and 'pakistan'), that was not the expected result for the given query. Since SIRF is dependent on dependency parser for natural language processing, its performance can be affected if the dependency parser produces a wrong parse. The dependency parser may produce a wrong parse if the user query does not have a proper grammatical structure. This issue can be resolved by auto-correcting user queries for grammatical mistakes. Since SIRF can be best utilised being integrated with other search tools, the task of auto-correction can be delegated to these tools.
- About 39% of the failed queries were caused by mapping failure for synonyms.
- The remaining 24% of the failed queries were caused by unused concepts. For example *price* is defined in DBPedia ontology but not used for *Books* and when a query asked books' prices, it does not bring any results.

10.10.2 Linking back to the Research Hypothesis

The above evaluation proves the research hypothesis (described in Chapter 1) that it is possible to design an effective natural language semantic search system using existing semantic technologies (i.e. RDF, Ontologies and SPARQL) to access linked open datasets. However, there are still some areas which require further work e.g. concept mapping and keywords mapping.

In addition to the problems that caused failed queries, the author also identified some limitations with the results retrieved from LODs i.e.

- Since SIRF had a non-commercial use of the Linked Open Data, it faced usage limits. The SPARQL endpoints for British Library and DBPedia has a limit on number of requests. These endpoints stop responding for a few minutes after this limit is over.
- The endpoints also have limits on the number of results returned as a response to a query. That means, despite getting results the system may not be getting a complete list of results.
- The author also noticed that some queries bring redundant results. Exploring further, the author identified that the redundant results are caused by redundant data in the dataset. The redundant data caused by variation of names and labels e.g. name for a person is saved twice as 'Julian Lincoln, Simon' and 'Simon, Julian Lincoln'.

10.11 Summary

This chapter has described the construction of a prototype to validate the functionality of SIRF. The chapter has explained the implementation of individual modules of the framework. The chapter has further evaluated the performance of existing semantic search NLIs (i.e. QAKiS and FREyA) and the proposed system (SIRF) based on the chosen criteria i.e. Accessibility, Portability, Extensibility, Interoperability and Result Optimisation.

The next chapter provides an overall summary of this thesis.

Chapter 11

Conclusions and Future Work

11.1 Introduction

The main motivation behind this thesis has been to improve the accessibility of semantic data so that non-technical end users and other search tools should be able to access published RDF datasets. The chapter summarises the contributions of this thesis to the field of knowledge and explores the possible future directions from this line of investigation.

11.2 Research Summary

The author investigated following research questions (as described Chapter 1)

11.2.1 What are the limitations of existing semantic search tools?

The author analysed available documentations and architecture of existing semantic search systems (Chapter 4) followed by evaluation of the selected systems i.e. FREyA and QAKiS (Chapter 10). Despite of the all good work done, FREyA and QAKiS were found to have a number of individual limitations e.g.

- FREyA uses clarifications dialogues to understand the intent of a user query and cannot answer a query unless it finds the meanings that it understands.
- QAKiS does not return any answer for a majority of the queries since it needs at least one named entity in the query that it can recognise.
- Both systems currently work on single datasets only, hence lack the feature of extensibility

- Both systems lack expressivity for natural language queries since the focus is to translate NL queries to SPARQL queries. These systems support only limited range of natural language queries.

11.2.2 What are the features an accessible search tool should have to eliminate these limitations?

The author identified the features (given below) for an accessible search tool to improve effectiveness of semantic search technologies for end users.

1. The first identified feature is 'accessibility' that refers to the convenience with which end users or software agents can access semantic datasets. To improve the accessibility, the semantic search should support multiple expressions for a natural language query (i.e. free-text queries) and be able to translate these expressions to a valid SPARQL query. As discussed above, FREyA and QAKiS only offer limited support for natural language questions hence lack this feature.

2. The second identified feature is 'portability' that enables a search system to be used with a different dataset. FREyA and QAKiS both are categorised as portable semantic search systems but these are tied to single datasets.

3. Since semantic web search is meant to possibly explore all related datasets (to a query) across the web, a semantic search tool should be extensible to merge more than one datasets. FREyA and QAKis are portable but not extensible since each work on single dataset at a time.

4. The existing semantic search systems do not cater for any user friendly search features e.g. auto-correct, auto-suggest and auto-fill. The reason for not having these features is obvious that their focus is currently on query translation and these features can be added later on. But the question is, do we really need to develop a separate semantic search engine? The author envisions semantic search as an extension to the existing commercial search tools, where the syntax-based search tool does all the natural language corrections and passes the sanitized natural language query to the semantic search tool. To achieve this, the semantic search tool should be interoperable with other search tools.

5. The semantic web identifies resources as unique HTTP identifiers. Therefore the SPARQL endpoints return results in form of URIs e.g. http://example.com/resource. Such results are not very readable or user friendly. To present results in a readable format, there is a need to parse URIs to their relevant text. For example,

the URI *http://dbpedia.org/page/Paul_McCartney* could be displayed as *Sir James Paul McCartney*. The author identifies result optimisation as a required feature for a semantic search tool.

11.2.3 Using the Research Questions above, is it feasible to design an accessible semantic search system that can be integrated with other search tools?

This thesis has introduced SIRF (a new framework for semantic search). SIRF seeks to overcome the limitations of existing search tools and provides a query interface that supports natural language queries. The main features of the framework are as follows:

(A) Query translation

From the end user point of view, semantic data is difficult to access given that an end user may not have the required technical knowledge to express their needs in a relatively complex query language, such as SPARQL. This is why translating user queries is one of the biggest challenges for a semantic search tool. The framework provides a facility to translate from informal end user query to formal SPARQL query. The existing NLIs for semantic search lack the ability to process queries containing negation, conjunction and quantification. The Query Formatter module in the framework handles such queries using SPARQL 1.1 features such as EXISTS, logical AND (&&), logical OR (||) and LIMIT (as discussed in Chapter 8).

(B) Caching for Efficiency

The speed of a search system is one of the most important features. The semantic search systems are based on ontological concepts and a semantic search tool needs to reason on the domain ontologies used by a particular system. To fetch the ontologies from the cloud every time it is needed to map a user query can be a tedious task. It can even fail if an ontology server does not respond or response time is slow. To handle such a situation, SIRF caches mapped ontologies to have a local decision support system. SIRF also precomputes direct and indirect relations between two ontology concepts. This precomputation further improves performance of the proposed system. Caching ontologies speeds up the process though at the same time it needs extra management to keep the cache updated. Caching is a technique used by all large scale and heavy traffic systems and it is worth the management cost and space. SIRF also caches SPARQL queries to save query processing time for queries that have been already asked.

(C) Result Optimisation

The thesis has covered two main aspects of result optimisation i.e. (i) Result Ranking and (ii) Result Readability. SIRF adopted ranking based on popularity of namespaces and the number of instances for a particular class. For the result readability, the author proposed to define result schemas. For this work, the author has manually defined schemas for the concepts of interest e.g. Person, Book etc. The tests (from Chapter 10) have shown that using schemas has significantly improved the readability of results.

(D) Interoperability

SIRF can be embedded in other search tools since it operates on natural language queries and can be set up with any RDF datasets. The tests (from Chapter 10) have shown that the prototype worked better with well structured queries because the NLP parser works better on well structured sentences. Existing commercial search engines are rich with tools for natural language processing and have the capability to re-phrase a user query to a well structured form. SIRF integrated with such systems can provide much better performance.

11.3 Possible Implementations

SIRF can be used as a backbone of the search tool for any system using semantic web technologies i.e. Linked Open Data (LOD) and a SPARQL endpoint to access LOD data. The framework contains a number of modules that may work independently with little configuration i.e. ontology processor, query optimizer and API.

This flexibility enables SIRF to be used as a whole or as a part of other search tools. The algorithms introduced in this work can be implemented in multiple ways to fit in a semantic search system.

11.4 Limitations

Though the research has made a number of contributions to improve the accessibility of semantic search, a number of limitations would remain:

- This current research considered a single natural language. The prototype involved user queries in English only, hence all language conversion procedures are based on English syntax and grammar.

- Recently, a number of tools have been developed to map ontologies. However, yet there is no solution that is a clear success. SIRF handles ontology mapping using basic techniques i.e. *owl:sameAs* property. For a small test bed (containing datasets from two domains), this technique ran with no issues but it needs to be tested on a large scale.

- The performance of SIRF is dependent on the quality of the NLP parser.

11.5 Future Challenges

To deploy this framework beyond the laboratory, additional effort would be required in terms of improving support for performance, automated generation of result schema and an enhanced approach towards complex queries.

11.5.1 Natural Language

Natural language seems to be the best approach for a novice user but at the same time it may limit the expressivity of a system. There are a number of NLP tools available but most of them slow down the speed of the whole transaction. The author is interested to look further into how to handle the NLP integration, so as not to negatively affect the performance of the system.

11.5.2 Ontology Mapping

Multiple ontologies can be defined for the same domain. Similarly more than one ontology can be used to define a single dataset. A system like a search tool needs to handle hundreds of ontologies and datasets, thus requires support for ontology mapping. Also, a system can fail to map correctly if an ontology is not properly designed or has a lack of information.

11.5.3 Scalability

Scalability within the context of a search tool is the ability of a system to handle a rapidly growing amount of data. In addition to the Ontology Mapping (as described in Section 11.5.2), semantic data also faces issues of incompleteness, noisy data and redundant information (as discussed in Chapter 10).

11.6 Concluding Remarks

Natural language interfaces are essential for end users access to RDF datasets. This thesis aimed to improve accessibility of semantic data by developing a new framework (i.e. SIRF) which supports natural language queries. Based on a comprehensive analysis of the state-of-the-art of existing semantic search tools, the thesis developed criteria to be met by a search tool and designed a framework that met those requirements. The author developed a prototype as a proof of concept to evaluate the proposed framework. A number of tests were run on two Linked Open datasets. A number of features of the framework were highlighted and further challenges were identified.

The author is convinced that this work advances the current understanding and state-of-the-art of semantic search NLIs. The author believes that the proposed framework (or similar architectures) can substantially improve the accessibility of semantic data, and that this approach offers a mechanism to unlock the potential of the semantic web for everyone.

References

Ambati, B. R., Deoskar, T. and Steedman, M., 2014. Improving dependency parsers using combinatory categorial grammar. In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics.*, vol. 2, pp. 159–163. 68

Antoniou, G., Groth, P., Harmelen, F. V. and Hoekstra, R., 2012. *A Semantic Web Primer, 3rd Edition*. The MIT Press, 3^{rd} ed. 13

Antoniou, G. and Harmelen, F. v., 2008. *A Semantic Web Primer, (Cooperative Information Systems)*. The MIT Press, 2^{nd} ed. 2, 12

Antoniou, G. and Van Harmelen, F., 2004. Web ontology language: OWL. In: *Handbook on ontologies*, Springer, pp. 67–92. 87

Antonsson, E. K., 1987. Development and testing of hypotheses in engineering design research. *Journal of Mechanisms, Transmissions, and Automation in Design* 109(2), pp. 153–154. 25

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.*, ACM, pp. 117–127. 107

Bard, K. B. and Ferrara, N., 2011. *Qualitative search engine based on factors of consumer trust specification*, [Online]. US Patent App. 13/108,349. 52

Beall, J., 2008. The Weaknesses of Full-Text Searching. *The Journal of Academic Librarianship* 34(5), pp. 438 – 444. Available at:<http://www.sciencedirect.com/science/article/pii/S0099133308001067>. 1

Beall, J., 2010. Geographical research and the problem of variant place names in digitized books and other full-text resources. *Library Collections, Acquisitions, and Technical Services* 34(2-3), pp. 74–82. Available at:<http://www.sciencedirect.com/science/article/pii/S146490551000031X>. 2

Beel, J. and Gipp, B., 2009. Google Scholar's ranking algorithm: an introductory overview. In: *Proceedings of the 12th International Conference on Scientometrics and Informetrics (ISSI'09).*, vol. 1, pp. 230–241. 21, 52

Berners-Lee, T., 1998. *Semantic web road map*, [Online]. Available at:<http://www.w3.org/DesignIssues/Semantic.html>. 2

Berners-Lee, T., 1999. *Weaving the Web*. Harper. 2, 10

Berners-Lee, T., 2006. *Artificial Intelligence and the Semantic Web*, [Online]. Available at:<http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html#(14)>. [Accessed: 01 August 2015]. xiii, 11

Bernstein, A. and Kaufmann, E., 2006. GINO - A Guided Input Natural Language Ontology Editor. In: *The Semantic Web - ISWC 2006*, Springer, pp. 144–157. xiii, 30, 32

Bernstein, A., Kaufmann, E. and Kaiser, C., 2005. Querying the Semantic Web with Ginseng: A Guided Input Natural Language Search Engine. In: *15^{th} Workshop on Information Technologies and Systems, Las Vegas, Nov. 2005.*, pp. 112–126. 30, 32

Bernstein, A. and Kiefer, C., 2006. Imprecise rdql: towards generic retrieval in ontologies using similarity joins. In: *Proceedings of the 2006 ACM symposium on Applied computing.*, ACM, pp. 1684–1689. 17

Bizer, C., 2009. The emerging web of linked data. *Intelligent Systems, IEEE* 24(5), pp. 87–92. 2

Blessing, L. T. and Chakrabarti, A., 2009. *DRM: A Design Research Methodology*. Springer. xiii, xvii, 24, 25, 26, 28, 112

Broekstra, J. and Kampman, A., 2003. SeRQL: A Second Generation RDF Query Language. In: *Proc. SWAD-Europe Workshop on Semantic Web Storage and Retrieval.*, pp. 13–14. 17, 33

Broekstra, J. and Kampman, A., 2004. Serql: An rdf query and transformation language. In: *Submitted to the International Semantic Web Conference, ISWC.*, Citeseer, vol. 2004. 17

Buil-Aranda, C., Hogan, A., Umbrich, J. and Vandenbussche, P.-Y., 2013. Sparql web-querying infrastructure: Ready for action? In: *The Semantic Web–ISWC 2013*, Springer, pp. 277–293. 17

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N. and Hullender, G., 2005. Learning to rank using gradient descent. In: *Proceedings of the 22nd international conference on Machine learning.*, ACM, pp. 89–96. 104

Cabrio, E., Cojan, J., Aprosio, A. P., Magnini, B., Lavelli, A. and Gandon, F., 2012. QAKiS: An Open Domain QA System based on Relational Patterns. In: *11th International Semantic Web Conference ISWC 2012.*, Citeseer, p. 9. xiii, 35, 36, 38, 113

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.*, ACM, pp. 74–83. 114

Chapelle, O. and Keerthi, S. S., 2010. Efficient algorithms for ranking with svms. *Information Retrieval* 13(3), pp. 201–215. 104

Chen, D. and Manning, C. D., 2014. A fast and accurate dependency parser using neural networks. In: *Empirical Methods in Natural Language Processing (EMNLP)*., vol. 1. 65

Cheung, K.-H., Frost, H. R., Marshall, M. S., Prud'hommeaux, E., Samwald, M., Zhao, J. and Paschke, A., 2009. A journey to semantic web query federation in the life sciences. *BMC bioinformatics* 10(10), p. 1. 38

Chopra, A., Prashar, A. and Sain, C., 2013. Natural language processing. *International Journal of Technology Enhancements and Emerging Engineering Research* 1(4), pp. 131–134. 64

Cimiano, P., Haase, P., Heizmann, J., Mantel, M. and Studer, R., 2008. Towards portable natural language interfaces to knowledge bases - The case of the ORAKEL system. *Data & Knowledge Engineering* 65(2), pp. 325–354. 30

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P., 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12, pp. 2493–2537. 65

Covington, M. A., 2001. A fundamental algorithm for dependency parsing. ACM, pp. 95–102. 68

Cunningham, H., 2002. GATE, a General Architecture for Text Engineering. *Computers and the Humanities* 36(2), pp. 223–254. 31

Damljanovic, D., Agatonovic, M. and Cunningham, H., 2010. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In: *The semantic web: Research and applications*, Springer, pp. 106–120. 34

Damljanovic, D., Agatonovic, M. and Cunningham, H., 2011. Freya: An interactive way of querying linked data using natural language. In: *The Semantic Web: ESWC 2011 Workshops.*, Springer, pp. 125–138. xiii, 3, 30, 34, 35

Damljanovic, D., Tablan, V. and Bontcheva, K., 2008. A Text-based Query Interface to OWL Ontologies. In: *Language Resources and Evaluation (LREC) 2008*. 3, 30, 33

Danica, D. and Bontcheva, K., 2009. Towards enhanced usability of natural language interfaces to knowledge bases. In: *Web 2.0 & Semantic Web*, Springer, pp. 105–133. 29, 31

De Virgilio, R., Guerra, F. and Velegrakis, Y., 2012. *Semantic Search over the Web*. Data-Centric Systems and Applications., Springer Berlin Heidelberg. Available at:<https://books.google.co.uk/books?id=aDTDXH0-i-oC>. 20

Desourdis, R., 2009. *Achieving Interoperability in Critical It and Communication Systems*. Artech House, Inc. 39

Dixon, B. E., Vreeman, D. J. and Grannis, S. J., 2014. The long road to semantic interoperability in support of public health: experiences from two states. *Journal of Biomedical Informatics* 49, pp. 3–8. 1

Du, Y. and Hai, Y., 2013. Semantic ranking of web pages based on formal concept analysis. *Journal of Systems and Software* 86(1), pp. 187–197. 52

DuCharme, B., 2013. *Learning Sparql*. O'Reilly Media, Inc. 17, 79

Duffy, A. and Andreasen, M., 1995. Enhancing the evolution of design science. In: *International Conference on Engineering Design.*, ICED'95. 25

Duffy, A. H. and O'Donnell, F., 1998. A design research approach. Critical Enthusiasm - Contributions to Design Science. In: *Workshop on Research Methods in AI in Design.*, pp. 80–116. xiii, 25

Eckert, C. M., Stacey, M. and Clarkson, P., 2003. The spiral of applied research: A methodological view on integrated design research. In: *Proceedings of the 14th International Conference on Engineering Design (ICED'03),*. 25

El-gayar, M. M., Mekky, N. and Atwan, A., 2015. Efficient Proposed Framework for Semantic Search Engine using New Semantic Ranking Algorithm. *International Journal of Advanced Computer Science and Applications(ijacsa)* 6(8). 1

Elbedweihy, K., Wrigley, S. N. and Ciravegna, F., 2012a. Evaluating semantic search query approaches with expert and casual users. In: *The Semantic Web–ISWC 2012*, Springer, pp. 274–286. 29, 30, 36

Elbedweihy, K., Wrigley, S. N. and Ciravegna, F., 2012b. Improving semantic search using query log analysis. *Interacting with Linked Data (ILD 2012)* p. 61. 37

Ell, B., Vrandečić, D. and Simperl, E., 2011. Deriving human-readable labels from sparql queries. In: *Proceedings of the 7th International Conference on Semantic Systems.*, ACM, pp. 126–133. 20

Ermilov, I., Martin, M., Lehmann, J. and Auer, S., 2013. Linked open data statistics: Collection and exploitation. In: *Knowledge Engineering and the Semantic Web*, Springer, pp. 242–249. 112

Faria, D., Pesquita, C., Santos, E., Cruz, I. F. and Couto, F. M., 2014. AgreementMakerLight: A scalable automated ontology matching system. *DILS 2014* p. 29. 14

Fatima, A., Luca, C. and Wilson, G., 2014a. New Framework for Semantic Search Engine. In: *UKSim-AMSS 16th International Conference on Computer Modelling and Simulation.*, pp. 446–451. 42, 45

Fatima, A., Luca, C. and Wilson, G., 2014b. User experience and efficiency for semantic search engine. In: *International Conference on Optimization of Electrical and Electronic Equipment (OPTIM), 2014.*, pp. 924–929. 45, 56

Fatima, A., Luca, C. and Wilson, G., 2015a. User queries for semantic search. *International Journal of Simulations, Systems, Science and Technology* 16(In Press). 42

Fatima, A., Luca, C., Wilson, G. and Kettouch, M., 2015b. Result optimisation for federated SPARQL queries. In: *UKSim-AMSS 17th International Conference on Computer Modelling and Simulation.*, IEEE. xvii, 45, 53, 103, 105, 106, 110

Ferré, S., 2014. Expressive and scalable query-based faceted search over SPARQL endpoints. In: *The Semantic Web–ISWC 2014*, Springer, pp. 438–453. 38, 150

Finlayson, M. A., 2014. Java Libraries for Accessing the Princeton WordNet: Comparison and Evaluation. In: *Proceedings of the 7th Global Wordnet Conference, Tartu, Estonia.* 114

Frank, A., Krieger, H.-U., Xu, F., Uszkoreit, H., Crysmann, B., Jörg, B. and Schäfer, U., 2007. Question answering from structured knowledge sources. *Journal of Applied Logic* 5(1), pp. 20–48. 3

Freitas, A. and Curry, E., 2014. Natural language queries over heterogeneous linked data graphs: A distributional-compositional semantics approach. In: *Proceedings of the 19th international conference on Intelligent User Interfaces.*, ACM, pp. 279–288. 38

García, J. M., Junghans, M., Ruiz, D., Agarwal, S. and Ruiz-Cortés, A., 2013. Integrating semantic web services ranking mechanisms using a common preference model. *Knowledge-Based Systems* 49, pp. 22–36. 104

Gerber, D., Hellmann, S., Bühmann, L., Soru, T., Usbeck, R. and Ngomo, A.-C. N., 2013. Real-time RDF extraction from unstructured data streams. In: *The Semantic Web-ISWC 2013*, Springer, pp. 135–150. 47

Gómez-Pérez, A. and Corcho, O., 2002. Ontology languages for the semantic web. *Intelligent Systems, IEEE* 17(1), pp. 54–60. 16

Gruber, T. R., 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition* 5(2), pp. 199–220. Available at:<http://dx.doi.org/10.1006/knac.1993.1008>. 14

Guarino, N., 1998. Formal ontology in information systems. In: *Proceedings of the 1st International Conference on Formal Ontology in Information Systems(FOIS'98), June 6-8, Trento, Italy.*, IOS press, vol. 46. 15

Haase, P., Broekstra, J., Eberhart, A. and Volz, R., 2004. A comparison of rdf query languages. In: *The Semantic Web–ISWC 2004*, Springer, pp. 502–517. 16, 17

Hales, C. *et al.*, 2003. Respect design or expect disaster! In: *DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm*. 24

Harris, S., Seaborne, A. and Prud'hommeaux, E., 2013. SPARQL 1.1 query language. *W3C Recommendation* 21. 79

He, X., Hua, E., Liu, X. and Lin, Y., 2011. *Computer, Informatics, Cybernetics and Applications: Proceedings of the CICA 2011*, vol. 107. Springer Science & Business Media. 12

Hevner, A. R., 2007. A three cycle view of design science research. *Scandinavian journal of information systems* 19(2), p. 4. 24

Hotho, A., Jäschke, R., Schmitz, C. and Stumme, G., 2006. *Information retrieval in folksonomies: Search and ranking*. Springer. 104

Jain, P., Verma, K., Yeh, P. Z., Hitzler, P. and Sheth, A. P., 2010. Loqus: Linked open data sparql querying system . 19

Jansen, B. J., Booth, D. L. and Spink, A., 2008. Determining the informational, navigational, and transactional intent of web queries. *Information Processing & Management* 44(3), pp. 1251–1266. 18

Jindal, V., Bawa, S. and Batra, S., 2014. A review of ranking approaches for semantic search on Web. *Information Processing & Management* 50(2), pp. 416–425. 107

Jurczyk, E., 2014. Using a customized search engine to address low health literacy: A program description. *Journal of the Canadian Health Libraries Association/Journal de l'Association des bibliothèques de la santé du Canada* 33(2), pp. 112–118. 2

Kakkonen, T., 2007. *Framework and resources for natural language parser evaluation*. PhD Dissertation, Department of Computer Science and Statistics, University of Joensuu, Finland. 65

Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D. and Scholl, M., 2002. Rql: a declarative query language for rdf. In: *Proceedings of the 11th international conference on World Wide Web.*, ACM, pp. 592–603. 16, 17

Karvounarakis, G., Magganaraki, A., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M. and Tolle, K., 2003. Querying the semantic web with rql. *Computer networks* 42(5), pp. 617–640. 16

Katz, B., Lin, J. and Quan, D., 2002. Natural language annotations for the semantic web. In: *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, Springer, pp. 1317–1331. 3

Kaufmann, E. and Bernstein, A., 2007. *How Useful are Natural Language Interfaces to the Semantic Web for Casual End-users?* 3, 30, 36

Kaufmann, E. and Bernstein, A., 2010. Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. *Web Semantics: Science, Services and Agents on the World Wide Web* 8(4), pp. 377–393. 29

Kaufmann, E., Bernstein, A. and Fischer, L., 2007. Nlp-reduce: A naive but domain-independent natural language interface for querying ontologies. ESWC Zurich. 30

Kaufmann, E., Bernstein, A. and Zumstein, R., 2006. Querix: A Natural Language Interface to Query Ontologies Based on Clarification Dialogs. Citeseer. 30, 32

Killoran, J., 2013. How to use search engine optimization techniques to increase website visibility. *Professional Communication, IEEE Transactions on* 56(1), pp. 50–66. 2

Klyne, G. and Carroll, J. J., 2006. Resource Description Framework (RDF): Concepts and Abstract Syntax. Tech. rep., Available at:<http://www.w3.org/TR/rdf-concepts/>. 2

Kothari, C., 2004. *Research methodology: Methods and techniques*. New Age International. 23

Langville, A. N., Meyer, C. D. and FernÁndez, P., 2008. GoogleâĂŹs pagerank and beyond: The science of search engine rankings. *The Mathematical Intelligencer* 30(1), pp. 68–69. 104

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S. *et al.*, 2014. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal* 5, pp. 1–29. 113

Li, H. and Wang, Y., 2010. Ranked keyword query on semantic web data. In: *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on.*, IEEE, vol. 5, pp. 2285–2289. 47

Linckels, S., 2008. *An e-librarian service: Supporting explorative learning by a description logics based semantic retrieval tool*. Ph.D. thesis. xiii, 33

Linckels, S. and Meinel, C., 2007. Semantic interpretation of natural language user input to improve search in multimedia knowledge base (semantische interpretation einer benutzer-eingabe in natürlicher sprache für eine verbesserte suche in einer multimedialen wissensdatenbank). *it-Information Technology* 49(1), pp. 40–48. 33

Lopez, V., Fernández, M., Motta, E. and Stieler, N., 2011. PowerAqua: Supporting users in querying and exploring the semantic web. *Semantic Web* 3(3), pp. 249–265. xiii, 3, 30, 34, 113

Lopez, V., Motta, E. and Uren, V., 2006. *PowerAqua: Fishing the semantic web*. Springer. 38

Lopez, V., Pasin, M. and Motta, E., 2005. AquaLog: An ontology-portable question answering system for the semantic web. In: *The Semantic Web: Research and Applications*, Springer, pp. 546–562. 30, 31

Lopez, V., Uren, V., Motta, E. and Pasin, M., 2007. AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics: Science, Services and Agents on the World Wide Web* 5(2), pp. 72–105. Available at:<http://www.sciencedirect.com/science/article/pii/S1570826807000145>. Software Engineering and the Semantic Web. xiii, 31

Malik, S. K. and Rizvi, S., 2012. A framework for SPARQL query processing, optimization and execution with illustrations. *International Journal of Computer Information Systems and Industrial Management Applications* 4, pp. 208–218. 49

Manshadi, M. and Li, X., 2009. Semantic tagging of web search queries. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP.*, Association for Computational Linguistics, vol. 2, pp. 861–869. 44

March, S. T. and Smith, G. F., 1995. Design and natural science research on information technology. *Decision Support Systems* 15(4), pp. 251 – 266. Available at:<http://www.sciencedirect.com/science/article/pii/0167923694000412>. 112

Masse, M., 2011. *REST API design rulebook*. O'Reilly Media, Inc. 146

McClosky, D., Che, W., Recasens, M., Wang, M., Socher, R. and Manning, C., 2012. Stanford's System for Parsing the English Web. In: *Workshop on the Syntactic Analysis of Non-Canonical Language (SANCL 2012). Montreal, Canada.* 68

Mehdi, M., Iqbal, A., Hogan, A., Hasnain, A., Khan, Y., Decker, S. and Sahay, R., 2014. Discovering domain-specific public SPARQL endpoints: A life-sciences use-case. In: *Proceedings of the 18th International Database Engineering & Applications Symposium.*, ACM, pp. 39–45. 17

Mikheev, A., Moens, M. and Grover, C., 1999. Named entity recognition without gazetteers. In: *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics.*, Association for Computational Linguistics, pp. 1–8. EACL'99. 44

Miller, G. A., 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38(11), pp. 39–41. 31

Miller, L., Seaborne, A. and Reggiori, A., 2002. Three implementations of squishql, a simple rdf query language. In: *International Semantic Web Conference.*, Springer, vol. 2342, pp. 423–435. 17

Minnaert, G., Sawyer, A. and Thayer, A., 2002. *Internet-based application program interface (api) documentation interface*, [Online]. US Patent 6,405,216. 48

Nadeau, D. and Sekine, S., 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30(1), pp. 3–26. 67

Naseer, O., Naseer, A., Khan, A. A. and Naseer, H., 2013. Using page size for controlling duplicate query results in semantic web. *International Journal of Web & Semantic Technology* 4(2), p. 49. 44

Nica, A., Suchanek, F. M. and Varde, A. S., 2015. New research directions in knowledge discovery and allied spheres. *ACM SIGKDD Explorations Newsletter* 16(2), pp. 46–49. 10

Oates, B. J., 2006. *Researching Information Systems and Computing*. Sage. xiii, 23, 24

Osman, T., Thakker, D. and Schaefer, G., 2010. Semantics-Based Intelligent Indexing and Retrieval of Digital Images - A Case Study. In: *Emergent Web Intelligence: Advanced Information Retrieval*, Springer, pp. 117–134. 16

Owen, C. L., 1998. Design research: Building the knowledge base. *Design Studies* 19(1), pp. 9–20. 25

Page, L., Brin, S., Motwani, R. and Winograd, T., 1999. The pagerank citation ranking: bringing order to the web. . 104

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 5th ACM international workshop on Web information and data management.*, ACM, pp. 58–61. 104

Patton, E. W., Seyed, P., Wang, P., Fu, L., Dein, F. J., Bristol, R. S. and McGuinness, D. L., 2014. Semanteco: A semantically powered modular architecture for integrating distributed environmental and ecological data. *Future Generation Computer Systems* 36, pp. 430–440. 38

Peffers, K., Tuunanen, T., Rothenberger, M. A. and Chatterjee, S., 2007. A design science research methodology for information systems research. *Journal of management information systems* 24(3), pp. 45–77. 24, 25

Petcu, D., 2011. Portability and interoperability between clouds: challenges and case study. In: *Towards a Service-Based Internet*, Springer, pp. 62–74. 39

Popescu, A.-M., Etzioni, O. and Kautz, H., 2003. Towards a theory of natural language interfaces to databases. In: *Proceedings of the 8th international conference on Intelligent user interfaces.*, ACM, pp. 149–157. 3

Proctor, I. A. R., Yang, M. and Zhao, H., 2014. *Executing server side script code specified using PHP on a server to generate dynamic web pages*, [Online]. US Patent 8,707,161. 115

Rakhmawati, N. A., Umbrich, J., Karnstedt, M., Hasnain, A. and Hausenblas, M., 2013. Querying over Federated SPARQL Endpoints - A State of the Art Survey. *arXiv preprint arXiv:1306.1723* . 136

Schmachtenberg, M., Bizer, C. and Paulheim, H., 2014. Adoption of the linked data best practices in different topical domains. In: *The Semantic Web–ISWC 2014*, Springer, pp. 245–260. xvii, 3, 11

Schweikardt, N., Schwentick, T. and Segoufin, L., 2010. Database theory: Query languages. In: *Algorithms and theory of computation handbook.*, Chapman & Hall/CRC, pp. 19–19. 16

Selvan, C. A., M.P. Sekar and Dharshini, P. A., 2012. Survey on web page ranking algorithms. *International Journal of Computer Applications* 41(19), pp. 5646–7764. Available at:<http://dx.doi.org/10.1016/S0169-023X(97)00056-6>. 2

Selvan, M. P., Sekar, A. C. and Dharshini, A. P., 2012. Survey on web page ranking algorithms. *International Journal of Computer Applications* 41(19), pp. 1–7. 47

Sharef, N. M. and Noah, S. A., 2013. Natural language query translation for semantic search. *International Journal of Digital Content Technology and its Applications* 7(13), p. 53. 64

Sharef, N. M. and Noah, S. A. M., 2012. Semantic search processing in natural language interface. In: *Computing and Convergence Technology (ICCCT), 2012 7th International Conference on.*, IEEE, pp. 1436–1442. 30, 36, 37

Shvaiko, P. and Euzenat, J., 2013. Ontology matching: state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on* 25(1), pp. 158–176. 14

Simon, H. A., 1996. *The sciences of the artificial*. MIT press. 25

Staab, S. and Studer, R., 2013. *Handbook on ontologies*. Springer Science & Business Media. 14

Stojanovic, N., 2003. On analysing query ambiguity for query refinement: The librarian agent approach. In: *Conceptual Modeling-ER 2003*, Springer, pp. 490–505. 21

Studer, R., Benjamins, V. R. and Fensel, D., 1998. Knowledge Engineering: Principles and Methods. *Data & Knowledge Engineering* 25(1-2), pp. 161–197. Available at:<http://dx.doi.org/10.1016/S0169-023X(97)00056-6>. 10

Tablan, V., Damljanovic, D. and Bontcheva, K., 2008. *A natural language query interface to structured information*. Springer. 36

Thompson, C. W., Pazandak, P. and Tennant, H. R., 2005. Talk to your semantic web. *IEEE Internet Computing* (6), pp. 75–78. 3

Tyckoson, D. and Dove, J., 2014. *Reimagining Reference in the 21st Century*. Charleston insights in library, archival, and information sciences. Available at:<https://books.google.co.uk/books?id=NxJZBQAAQBAJ>. 36

Upadhyaya, B. P., 2014. REST client pattern. In: *Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on.*, IEEE, pp. 231–235. 116

Uren, V., Lei, Y., Lopez, V., Liu, H., Motta, E. and Giordanino, M., 2007. The usability of semantic search tools: A review. *The Knowledge Engineering Review* 22(04), pp. 361–377. 29, 38

Usbeck, R., Ngomo, A.-C. N., Bühmann, L. and Unger, C., 2015. HAWK - Hybrid Question Answering Using Linked Data. In: *The Semantic Web. Latest Advances and New Domains*, Springer, pp. 353–368. 38, 39

Vaishnavi, V. and Kuechler, W., 2004. Design research in information systems . 25

Vaishnavi, V., Kuechler, W. and Kuechler, B., 2013. *Design science research in information systems*, [Online]. Available at:<http://desrist.org/desrist/content/design-science-research-in-information-systems.pdf>. 24

Van Heijst, G., Schreiber, A. T. and Wielinga, B. J., 1997. Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies* 46(2), pp. 183–292. 15

Vijayadeepa, V. and Ghosh, D., 2013. Sem-Rank: A Page Rank Algorithm Based on Semantic Relevancy for Efficient Web Search. *International Review on Computers and Software (IRECOS)* 8(11), pp. 2642–2647. 107

Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I. and Hinton, G., 2014. Grammar as a foreign language. *arXiv preprint arXiv:1412.7449* . 68

W3C., 2011. *Semantic web*, [Online]. Available at:<http://www.w3.org/conf/2011/>. 11

Wei, W., Barnaghi, P. and Bargiela, A., 2008. Search with meanings: An overview of semantic search systems 3, pp. 76–82. 29

Wei, W., Barnaghi, P. and Bargiela, A., 2011. Rational research model for ranking semantic entities. *Information Sciences* 181(13), pp. 2823–2840. 107

Ye, J., Dasiopoulou, S., Stevenson, G., Meditskos, G., Kontopoulos, E., Kompatsiaris, I. and Dobson, S., 2015. Semantic web technologies in pervasive computing: A survey and research roadmap. *Pervasive and Mobile Computing (Accepted for publication)* . 16

Zhang, Y., Panangadan, A. and Prasanna, V. K., 2015. Ufomq: An algorithm for querying for similar individuals in heterogeneous ontologies. In: *Big Data Analytics and Knowledge Discovery*, Springer, pp. 178–189. 21

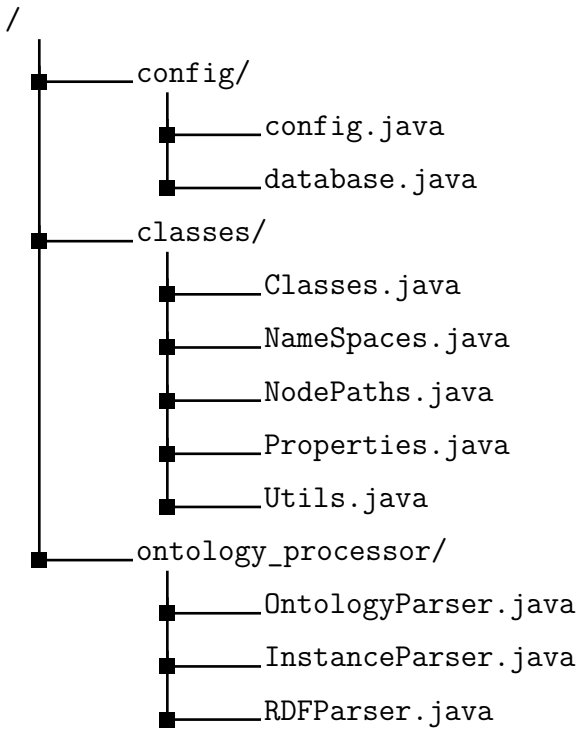
Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Chen, K. and Sun, G., 2008. A general boosting method and its application to learning ranking functions for web search. In: *Advances in neural information processing systems.*, pp. 1697–1704. 104

Zhou, Q., Wang, C., Xiong, M., Wang, H. and Yu, Y., 2007. *SPARK: adapting keyword query to semantic search*. Springer. 29

Appendix I - Implementation of Knowledge Base

This section provides the code for the implementation of the framework's modules that build its Knowledge Base (KB) i.e. the Data Parser (RDF Parser) and Ontology Processor.

Code Listing of KB (Prototype)



Listing 1: "config/config.java"

```
package config;

/**
 *
 * @author Arooj
 */
public class config {
    public static String DBNAME = "ir_db_test";
    public static String DBUSER = "root";
    public static String DBPASSWORD = "";
    public static String DBURL = "jdbc:mysql://localhost:3306/";
    public static String DBDRIVER = "com.mysql.jdbc.Driver";
    public static int domain_id = 1;
    public static String domain_short = "BL";

}
```

Listing 2: "config/database.java"

```
package config;

import com.mysql.jdbc.jdbc2.optional.MysqlDataSource;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

/**
 *
 * @author Arooj
 */
public class database {

    private static Connection con;
    private static final String Driver = config.DBDRIVER;
    private static final String Url = config.DBURL;
    private static final String db = config.DBNAME;
    private static final String user = config.DBUSER;
    private static final String pwd = config.DBPASSWORD;

    /**
     * create Database object
     */
    public database() {
    }

    /**
     * to load the database base driver
     *
     * @return a database connection
     * @throws SQLException throws an
     * exception if an error occurs
     */
    public static Connection loadDriver() throws SQLException {
        MysqlDataSource dataSource = new MysqlDataSource();
        dataSource.setUser(user);
        dataSource.setPassword(pwd);
        dataSource.setDatabaseName(db);
        dataSource.setServerName("localhost");
        con = dataSource.getConnection();
        return con;
    }
```

```
/**
 * to get a result set of a query
 *
 * @param query custom query
 * @return a result set of custom
 * query
 */
public static ArrayList getResultSet(String query) throws SQLException {
    ResultSet rs = null;
    PreparedStatement st = null;
    ArrayList<String> arrayList = null;
    try {
        con = loadDriver();
        st = con.prepareStatement(query);
        rs = st.executeQuery();
        int rowSize = rs.getRow();
        ResultSetMetaData rsmd = rs.getMetaData();
        int columnSize = rsmd.getColumnCount();
        arrayList = new ArrayList<>();
        while (rs.next()) {
            int i = 1;
            while (i <= columnSize) {
                arrayList.add(rs.getString(i++));
            }
            i++;
        }
    } catch (SQLException ex) {
        System.out.println("Database->SQL exec error:" + query);
        System.out.println(ex.getMessage());
        System.exit(0);
    } finally {
        if (rs != null) {
            rs.close();
        }
        if (st != null) {
            st.close();
        }
        if (con != null) {
            con.close();
        }
    }
    return arrayList;
}

/**
 * to run an update query such as
 * update, delete
 *
 * @param query custom query
 * @return
 */
public static int runQuery(String query) throws SQLException {
    int id = 0;
    PreparedStatement st = null;
    ResultSet rs = null;
    try {
        con = loadDriver();
        st = con.prepareStatement(query, Statement.RETURN_GENERATED_KEYS);
        id = st.executeUpdate();
        rs = st.getGeneratedKeys();
        while (rs.next()) {
            id = rs.getInt(1);
        }
    } catch (SQLException ex) {
        System.out.println("Database->runQuery error:" + query);
        System.out.println(ex.getMessage());
        System.exit(0);
    } finally {
        if (rs != null) {
            rs.close();
        }
        if (st != null) {
            st.close();
```

```
        }
        if (con != null) {
            con.close();
        }
    }
    return id;
}

/**
 *
 * @param table table name
 * @param fields fields you want to
 * be returned back
 * @param where where clause pass
 * condition without where keyword
 * @return rows of result set
 */
public static ArrayList getResults(String table, String[] fields, String where) {
    ArrayList<String> data = null;
    String query = "SELECT ";
    int i;
    int fcounts = fields.length;
    try {
        if (table == null || table.equals("")) {
            throw new Exception("Please enter table name");
        }

        if (fcounts == 0) {
            throw new Exception("Please enter at least one field name");
        }

        for (i = 0; i < fcounts; i++) {
            if (i > 0) {
                query += ",";
            }
            query += fields[i];
        }
        query += " FROM " + table;
        if (where != null && !"".equals(where)) {
            query += " WHERE " + where;
        }
        data = database.getResultSet(query);
    } catch (Exception ex) {
        System.out.println("Database->get results error:" + query);
        System.out.println(ex.getMessage());
        return data;
        // System.exit(0);
    }
    return data;
}

/**
 * @param table table name
 * @param fields table fields you
 * want to add data
 * @param values values (row column)
 * in same order as fields
 * @return ready to execute insert
 * query
 */
public static int query(String table, String[] fields, String[] values) throws SQLException {
    String query = "";
    int last_insert = 0;
    PreparedStatement st = null;
    int pCount = fields.length;
    ResultSet rs = null;
    try {
        con = loadDriver();
        if ("".equals(table)) {
            throw new Exception("Error: Please enter database table name.");
        }
        int colCount = fields.length;
        int rCount = values.length;
```

```
            String vals = "";
            String separator = "";
            query += "INSERT IGNORE INTO " + table + " (";
            for (int i = 0; i < colCount; i++) {
                if (i > 0) {
                    query += ", ";
                    separator = ", ";
                }
                query += fields[i];
                vals += separator + " ? ";
            }
            query += ") values (" + vals + ") ";
            st = con.prepareStatement(query, Statement.RETURN_GENERATED_KEYS);
            for (int i = 1; i <= pCount; i++) {
                st.setString(i, values[i - 1]);
            }
            st.executeUpdate();
            rs = st.getGeneratedKeys();
            // rs = st.executeQuery();
            if (rs.first()) {
                last_insert = rs.getInt(1);
            }
        } catch (Exception ex) {
            System.out.println("Database->query invalid parameter:");
            System.out.println(ex.getMessage());
            System.exit(0);
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (st != null) {
                st.close();
            }
            if (con != null) {
                con.close();
            }
        }
        return last_insert;
    }

    public static int getResultsWithParameters(String query, String[] params) throws SQLException {
        ResultSet rs = null;
        int id = 0;
        int pCount = params.length;
        PreparedStatement st = null;
        try {
            con = loadDriver();
            if (query == null || query.equals("")) {
                throw new Exception("Please enter table name");
            }
            st = con.prepareStatement(query);
            for (int i = 1; i <= pCount; i++) {
                st.setString(i, params[i - 1]);
            }
            rs = st.executeQuery();
            if (rs.first()) {
                id = rs.getInt(1);
            }
        } catch (SQLException ex) {
            System.out.println("Database->SQL exec error:" + query);
            System.out.println(ex.getMessage());
            System.exit(0);
        } catch (Exception ex) {
            System.out.println("Database->SQL exec error:" + query);
            System.out.println(ex.getMessage());
            System.exit(0);
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (st != null) {
                st.close();
            }
```

```
            if (con != null) {
                con.close();
            }
        }
        return id;
    }
}
```

Listing 3: "classes/Classes.java"

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package classes;

import com.hp.hpl.jena.ontology.OntClass;
import config.database;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Iterator;
import static ontology_processor.InstanceParser.domain_id;

/**
 *
 * @author Arooj
 */
public class Classes {
    private final static String table_classes = "classes";
    private final static String[] fields_classes = {"id", "class", "title", "`schema`", "group_id"};
    private final static String table_class_domain = "class_domains";
    private final static String[] fields_class_domain = {"id", "class_id", "domain_id", "note"};
    private final static String table_sub_classes = "sub_classes";
    private final static String[] fields_sub_classes = {"id", "class", "sub_class"};
    private final static String table_class_group = "class_groups";
    private final static String[] fields_class_group = {"id", "hint"};
    private final static String table_class_map = "class_map";
    private final static String[] fields_class_map = {"id", "class", "term"};

    public static void saveClass(OntClass thisClass, int domain_id, String domain_short, String ns_short) throws
         SQLException{
        String[] save_data = {"0",thisClass.toString(), thisClass.getLocalName(), ns_short, "0"};
        String where_c = " class=\""+thisClass.toString()+"\" ";
        if(thisClass.getLocalName() != null && !(thisClass.getLocalName().isEmpty())){
            // don't enter class if it does not have a title
            ArrayList data_c = database.getResults(table_classes, fields_classes, where_c);
            if(data_c == null || data_c.isEmpty()){
                int class_id = database.query(table_classes, fields_classes, save_data);
                // save class for domain
                String[] save_class_domain = {"0", String.valueOf(class_id), String.valueOf(domain_id),
                    domain_short+"-"+thisClass.getLocalName()};
                database.query(table_class_domain, fields_class_domain, save_class_domain);
            }
        }
    }
    public static int createGroup(String class_title) throws SQLException{
        String[] save_data = {"0",class_title};
        int class_id = database.query(table_class_group, fields_class_group, save_data);
        return class_id;
    }
    public static void setEquivalentClasses(OntClass thisClass, String ns_short) throws SQLException{
        String in_statement = "\""+thisClass+"\"";
        for (Iterator iit = thisClass.listEquivalentClasses(); iit.hasNext(); ) {
            OntClass c = (OntClass) iit.next();
            if(!(c.getLocalName()==null) && Utils.isValidWord(c.getLocalName().toLowerCase())){
                Inflector inf = new Inflector();
                String plural = inf.pluralize(c.getLocalName().toLowerCase());
                System.out.println("Class Found :  "+c.toString()+"--"+plural );
```

```
            String[] save_class_map_singular = {"0", c.toString(), c.getLocalName().toLowerCase()};
            database.query(table_class_map, fields_class_map, save_class_map_singular);

            String[] save_class_map_plural = {"0", c.toString(), plural};
            database.query(table_class_map, fields_class_map, save_class_map_plural);
        }
        in_statement += ", \""+c.toString()+"\"";
    }
    String where = " group_id > 0 AND class IN ("+in_statement+") LIMIT 1 ";
    ArrayList data = database.getResults(table_classes, fields_classes, where);
    int group_id = 0;
    if(data == null || data.isEmpty()){
        group_id = Classes.createGroup(thisClass.getLocalName());
    }else{
        try{
            group_id = Integer.parseInt(data.get(3).toString());
        }
        catch(NumberFormatException n){
            group_id = 0;
        }
    }
    // update group_id for the class
    String class_query = "UPDATE classes SET group_id = \""+group_id+"\" WHERE class=\""+thisClass.toString()+"\" ";
    database.runQuery(class_query);
    // run loop again to dump data
    for (Iterator iit = thisClass.listEquivalentClasses(); iit.hasNext(); ) {
        OntClass c = (OntClass) iit.next();
        String where_c = " class=\""+c.toString()+"\" ";
        if(c.getLocalName() != null && !(c.getLocalName().isEmpty())){
            ArrayList data_c = database.getResults(table_classes, fields_classes, where_c);
            if(data_c == null || data_c.isEmpty()){
                String[] save_data_eq = {"0",c.toString(),c.getLocalName(), ns_short, String.valueOf(group_id) };
                database.query(table_classes, fields_classes, save_data_eq);
            }else{
                String eq_class_query = "UPDATE classes SET group_id = \""+group_id+"\" WHERE
                     class=\""+c.toString()+"\" ";
                database.runQuery(eq_class_query);
            }
        }
    }
}
public static void setSubClasses(OntClass thisClass) throws SQLException{
    for (Iterator iit = thisClass.listSubClasses(); iit.hasNext(); ) {
        try{
            OntClass c = (OntClass) iit.next();
            String[] save_data_sub = {"0",thisClass.toString(),c.toString() };
            String where_c = " class=\""+thisClass.toString()+"\" AND sub_class = \""+c.toString()+"\" ";
            ArrayList data_c = database.getResults(table_sub_classes, fields_sub_classes, where_c);
            if(data_c == null || data_c.isEmpty()){
                if(Utils.validateHTTP_URI(c.toString()) && Utils.validateHTTP_URI(thisClass.toString())){
                    database.query(table_sub_classes, fields_sub_classes, save_data_sub);
                }
            }
        }
        catch(NullPointerException e){
            System.out.println("Exception occured for sub class "+e.getMessage());
        }
    }
}
 public static void setSuperClasses(OntClass thisClass) throws SQLException{
    for (Iterator iit = thisClass.listSuperClasses(); iit.hasNext(); ) {
        try{
            OntClass c = (OntClass) iit.next();
            String[] save_data_sub = {"0",c.toString(),thisClass.toString() };
            String where_c = " class=\""+c.toString()+"\" AND sub_class = \""+thisClass.toString()+"\" ";
            ArrayList data_c = database.getResults(table_sub_classes, fields_sub_classes, where_c);
            if(data_c == null || data_c.isEmpty()){
                if(Utils.validateHTTP_URI(c.toString()) && Utils.validateHTTP_URI(thisClass.toString())){
                    database.query(table_sub_classes, fields_sub_classes, save_data_sub);
                }
            }
        }
        catch(NullPointerException e){
```

```
                System.out.println("Exception occured for sub class "+e.getMessage());
            }
        }
    }
    public static String findChild(String classes) throws SQLException {
        String result = null;
        String sql = "SELECT `sub_class` FROM `sub_classes` WHERE `class` in ("+classes+")  and `sub_class` in
             ("+classes+") "
                + " ORDER BY `sub_class` ASC limit 0,1";
        ArrayList rs = database.getResultSet(sql);
        if(rs != null && !rs.isEmpty()){
            result = rs.get(0).toString();
        }
        return result;
    }
}
```

Listing 4: "classes/NameSpaces.java"

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package classes;

import com.hp.hpl.jena.ontology.OntModel;
import config.database;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Map;
import java.util.Random;

/**
 *
 * @author Arooj
 * The class has functions to save a name space in the database,
 * to find prefix for a namespace from the database
 */
public class NameSpaces {
    private final static String table_ontologies = "namespaces";
    private final static String[] fields_ontologies = {"id", "short", "namespace", "domain_id"};

    private final static String table_ranks = "ranks";
    private final static String[] fields_ranks = {"id", "namespace", "usage_counter"};

    public static String getNameSpaceShort(String uri, int domain_id) throws SQLException{
        String ns = "";
        ArrayList data = database.getResults(table_ontologies, new String[]{"short"}, " namespace=\""+uri+"\" AND
             domain_id='"+domain_id+"' ");
        if(data == null || data.isEmpty()){
            // if namespace not found in the database
            ns = NameSpaces.getSaltString();
            NameSpaces.saveNameSpace(uri, ns, domain_id);
            NameSpaces.saveRank(uri);
        }
        else{
             ns = data.get(0).toString();
        }
       return ns;
   }
   public static void saveRank(String uri) throws SQLException{
        String[] rank_ns = {"0",  uri, "usage_counter + 1"};
        database.query(table_ranks, fields_ranks, rank_ns);
   }
   public static void saveNameSpaces(OntModel model, int domain_id) throws SQLException{
        Map<String, String> nsPrefixMap = model.getNsPrefixMap();
        Iterator it = nsPrefixMap.entrySet().iterator();
        while (it.hasNext()) {
```

```
            Map.Entry pair = (Map.Entry)it.next();
            //System.out.println(pair.getKey() + " = " + pair.getValue());
            NameSpaces.saveNameSpace(pair.getValue().toString(), pair.getKey().toString(), domain_id);
            it.remove(); // avoids a ConcurrentModificationException
        }
    }
    public static void saveNameSpace(String uri, String ns_short, int domain_id) throws SQLException{
        // save class for domain
        String[] save_ns = {"0", ns_short, uri, String.valueOf(domain_id)};
        database.query(table_ontologies, fields_ontologies, save_ns);
    }
    protected static String getSaltString() {
        String SALTCHARS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
        StringBuilder salt = new StringBuilder();
        Random rnd = new Random();
        while (salt.length() < 6) {
            int index = (int) (rnd.nextFloat() * SALTCHARS.length());
            salt.append(SALTCHARS.charAt(index));
        }
        String saltStr = salt.toString();
        return saltStr;

    }
}
```

Listing 5: "classes/NodePaths.java"

```
package classes;

import java.util.ArrayList;
import java.util.HashMap;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

/**
 *
 * @author Arooj
 */
public class NodePaths {
    String firstNode;
    String nodeClass;
    String parentPath;
    ArrayList properties = new ArrayList<>();
    ArrayList nodeProperties = new ArrayList<>();
    ArrayList nodePaths = new ArrayList<>();
    NodeList nodeList;
    NodeList[] NodesLists;

    public NodePaths(NodeList nl){
       this.output("Node Paths Called");
       this.nodeList = nl;
    }

    private void output(String str ){
        System.out.println(str);
    }

    public void setFristNode(String str) {
        this.firstNode = str;
    }

    public void setParentPath(String str) {
        this.parentPath = str;
    }
    public void processNode(){
       this.getProperties(this.nodeList);
       if(this.parentPath != null) {
          this.showOutput();
       }
```

```
       this.processProperties();
    }

    public void getProperties(NodeList nl) {
        String str_class = null;
        HashMap<String,String> keyList= new HashMap<String,String>();

        int len = nl.getLength();
        for(int i=0; i<len; i++){
           Node tempNode = nl.item(i);
           if("rdf:type".equals(tempNode.getNodeName())){
               Element eElement = (Element) tempNode;
               String clsName = this.getClassName(eElement.getAttribute("rdf:resource"));
               if(!keyList.containsKey(clsName))
                str_class =  eElement.getAttribute("rdf:resource");

               if(this.parentPath == null)
                this.nodePaths.add(eElement.getAttribute("rdf:resource"));
               else
                this.nodePaths.add(this.parentPath + eElement.getAttribute("rdf:resource") + "}");
               keyList.put(clsName, eElement.getAttribute("rdf:resource"));

           } else if(tempNode.getChildNodes().getLength() > 1)  {
               if(!"rdf:Description".equals(tempNode.getNodeName())) {
                    this.properties.add(tempNode.getNodeName());
                    this.nodeProperties.add(tempNode.getChildNodes());
               }
           }
        }
        if(this.firstNode == null)
            this.firstNode = str_class;

        this.nodeClass = str_class;
    }

    public String getClassName(String url) {
        String str_class = null;
        String[] pcs = url.split("/");
        str_class = pcs[pcs.length-1];
        return str_class;
    }

    public void showOutput() {
        System.out.println("Paths: "+this.nodePaths);
    }

    void processProperties() {
        int count = 0;
        for (Object nodePropertie : this.nodeProperties) {
            NodeList tempNode = (NodeList) nodePropertie;
            NodePaths np = new NodePaths(tempNode.item(1).getChildNodes());
            np.setFristNode(this.firstNode);
            if(this.parentPath!=null)
                np.setParentPath(this.parentPath + this.nodeClass+"}{" + this.nodeClass + "[" +
                     this.properties.get(count)+"]");
            else
                np.setParentPath("{" + this.nodeClass + "[" + this.properties.get(count) + "]");
            np.processNode();
            count++;
        }

    }
}
```

Listing 6: "classes/Properties.java"

```
package classes;

import com.hp.hpl.jena.ontology.EnumeratedClass;
import com.hp.hpl.jena.ontology.OntClass;
```

```
import com.hp.hpl.jena.ontology.OntProperty;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.util.iterator.ExtendedIterator;
import config.config;
import config.database;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Iterator;

/**
 *
 * @author Arooj
 */
public class Properties {

    private final static String table_properties = "properties";
    private final static String[] fields_properties = {"id", "class", "property", "property_short", "domain",
         "property_range", "property_type", "property_group"};
    private final static String table_property_domain = "property_domains";
    private final static String[] fields_property_domain = {"id", "property_id", "domain_id", "note"};
    private final static String table_sub_properties = "sub_properties";
    private final static String[] fields_sub_properties = {"id", "property", "sub_property"};
    private final static String table_property_group = "properties_group";
    private final static String[] fields_property_group = {"id", "title"};
    private final static String table_property_map = "property_map";
    private final static String[] fields_property_map = {"id", "property","property_short", "term"};

    public static void saveProperty(OntClass thisClass, int domain_id, String domain_short) throws SQLException{
        ExtendedIterator<OntProperty> it = thisClass.listDeclaredProperties();
        int prop_id = 0;
            while(it.hasNext()){
                OntProperty p = it.next();
                String ptype="";
                if(p.isDatatypeProperty()) ptype = "0";
                else if(p.isObjectProperty()) ptype = "1";
                else ptype = "3";

                String propert="", dom="", rang="", ns="", ns_data = "";
                ns = NameSpaces.getNameSpaceShort(p.getNameSpace(), domain_id);
                if (p.isDatatypeProperty() && p.getDomain()!=null && p.getRange()!=null){
                    propert = p.toString();
                    dom  = p.getDomain().toString();
                    EnumeratedClass e = null;
                    ExtendedIterator<RDFNode> ii = null;
                    if(p.getRange().asClass().isEnumeratedClass()){
                        e = p.getRange().asClass().asEnumeratedClass();
                        ii = e.getOneOf().iterator();
                        RDFNode prop = null;
                        String s=null;
                        while(ii.hasNext()){
                            prop = ii.next();
                            s=prop.asLiteral().toString().split("\\^\\^")[0];
                            rang = s;
                        }
                    }else{
                        rang = p.getRange().toString();
                    }
                    String class_uri;
                    if(Utils.validateHTTP_URI(thisClass.toString())){
                        class_uri = thisClass.toString();
                    }
                    else{
                        class_uri = "";
                    }
                    // Save property
                    String where_c = " property=\""+p.toString()+"\" AND class=\""+class_uri+"\" AND domain=\""+dom+"\"
                         AND property_range=\""+rang+"\" ";
                    ArrayList data_c = database.getResults(table_properties, fields_properties, where_c);
                    if(data_c == null || data_c.isEmpty()){
                        String[] save_data_prop = {"0",class_uri, propert, ns+":"+p.getLocalName(), dom, rang, ptype,
                             "0" };
                        prop_id = database.query(table_properties, fields_properties, save_data_prop);
                        if(!(p.getLocalName()==null) && Utils.isValidWord(p.getLocalName().toLowerCase())){
```

```
                    Inflector inf = new Inflector();
                    String plural = inf.pluralize(p.getLocalName().toLowerCase());
                    String[] save_class_map_singular = {"0", p.toString(), ns+":"+p.getLocalName(),
                        p.getLocalName().toLowerCase()};
                    database.query(table_property_map, fields_property_map, save_class_map_singular);
                    String[] save_class_map_plural = {"0", p.toString(), ns+":"+p.getLocalName(), plural};
                    database.query(table_property_map, fields_property_map, save_class_map_plural);
                }
            }
            else{
            }
        }
        else{
            // Load non data properties depicting special relations using domain and range
            if(p.getDomain()==null && p.getRange()==null){

            }
            else{
                try{
                    String dom1, rang1;
                    dom1  = (p.getDomain()==null)?"":p.getDomain().toString();
                    rang1 = (p.getRange()==null)?"":p.getRange().toString();
                    String where_c = " property=\""+p.toString()+"\" ";
                    ArrayList data_c = database.getResults(table_properties, fields_properties, where_c);
                    if(data_c == null || data_c.isEmpty()){
                        String class_uri;
                        if(Utils.validateHTTP_URI(p.getClass().toString())){
                            class_uri = p.getClass().toString();
                        }
                        else{
                            class_uri = "";
                        }
                        String[] save_data_prop = {"0",class_uri, p.toString(), ns+":"+p.getLocalName(), dom1,
                            rang1, ptype, "0"};
                        prop_id = database.query(table_properties, fields_properties, save_data_prop);
                        if(!(p.getLocalName()==null) && Utils.isValidWord(p.getLocalName().toLowerCase())){
                            Inflector inf = new Inflector();
                            String plural = inf.pluralize(p.getLocalName().toLowerCase());
                            String[] save_class_map_singular = {"0", p.toString(), ns+":"+p.getLocalName(),
                                p.getLocalName().toLowerCase()};
                            database.query(table_property_map, fields_property_map, save_class_map_singular);
                            String[] save_class_map_plural = {"0", p.toString(), ns+":"+p.getLocalName(),
                                plural};
                            database.query(table_property_map, fields_property_map, save_class_map_plural);
                        }
                    }
                }
                catch(NullPointerException e){
                    System.out.println("Exception occured for property "+e.getMessage());
                }
            }
        }
        if(prop_id > 0){
            String where_c = " property_id=\"" + String.valueOf(prop_id) + "\" AND domain_id =
                \""+String.valueOf(domain_id) + "\" ";
            ArrayList data_c = database.getResults(table_property_domain, fields_property_domain, where_c);
            if(data_c == null || data_c.isEmpty()){
                // save property for domain
                String[] save_class_domain = {"0", String.valueOf(prop_id), String.valueOf(domain_id),
                    domain_short+ "-" + thisClass.getLocalName()};
                database.query(table_property_domain, fields_property_domain, save_class_domain);
                Properties.setEquivalentProperties(p);
                Properties.setSubProperties(p);
            }
        }
    } // end while declared properties
}
public static void setEquivalentProperties(OntProperty thisProp) throws SQLException{
    String in_statement = "\""+thisProp+"\"";
    for (Iterator iit = thisProp.listEquivalentProperties(); iit.hasNext(); ) {
        OntProperty c = (OntProperty) iit.next();
        in_statement += ", \""+c.toString()+"\"";
    }
```

```
    String where = " property_group > 0 AND property IN (" + in_statement + ") LIMIT 1 ";
    ArrayList data = database.getResults(table_properties, fields_properties, where);
    int group_id = 0;
    if(data == null || data.isEmpty()){
        group_id = Properties.createGroup(thisProp.getLocalName());
    }else{
        group_id = Integer.parseInt(data.get(7).toString());
    }
    // update group_id for the class
    String prop_query = "UPDATE properties SET property_group = \"" + group_id + "\" WHERE property=\"" +
         thisProp.toString() + "\" ";
    database.runQuery(prop_query);
    // run loop again to dump data
    for (Iterator iit = thisProp.listEquivalentProperties(); iit.hasNext(); ) {
        OntProperty c = (OntProperty) iit.next();
        String where_c = " property=\""+c.toString()+"\" ";
        ArrayList data_c = database.getResults(table_properties, fields_properties, where_c);
        if(data_c == null || data_c.isEmpty()){
            String dom1, rang1;
            dom1  = (c.getDomain()==null)?"":c.getDomain().toString();
            rang1 = (c.getRange()==null)?"":c.getRange().toString();
            String ptype = "";

            if(c.isDatatypeProperty()) ptype = "0";
            else if(c.isObjectProperty()) ptype = "1";
            else ptype = "3";
            String ns = NameSpaces.getNameSpaceShort(c.getNameSpace(), config.domain_id);
            String class_uri;
            if(Utils.validateHTTP_URI(c.getClass().toString())){
                class_uri = c.getClass().toString();
            }
            else{
                class_uri = "";
            }
            String[] save_data_eq = {"0", class_uri, c.toString(), ns+":"+c.getLocalName(), dom1, rang1, ptype,
                 String.valueOf(group_id) };
            database.query(table_properties, fields_properties, save_data_eq);
            if(!(c.getLocalName()==null) && Utils.isValidWord(c.getLocalName().toLowerCase())){
                        Inflector inf = new Inflector();
                        String plural = inf.pluralize(c.getLocalName().toLowerCase());

                        String[] save_class_map_singular = {"0", c.toString(), ns+":"+c.getLocalName(),
                             c.getLocalName().toLowerCase()};
                        database.query(table_property_map, fields_property_map, save_class_map_singular);

                        String[] save_class_map_plural = {"0", c.toString(), ns+":"+c.getLocalName(), plural};
                        database.query(table_property_map, fields_property_map, save_class_map_plural);
            }
        }else{
            String eq_class_query = "UPDATE properties SET property_group = \""+group_id+"\" WHERE
                 property=\""+c.toString()+"\" ";
            database.runQuery(eq_class_query);
        }
    }
}
public static void setSubProperties(OntProperty thisProp) throws SQLException{
    for (Iterator iit = thisProp.listSubProperties(); iit.hasNext(); ) {
        try{
            OntProperty c = (OntProperty) iit.next();
            if(thisProp.toString().equals(c.toString())){
                // do nothing
            }
            else{
                String[] save_data_sub = {"0",thisProp.toString(),c.toString() };
                database.query(table_sub_properties, fields_sub_properties, save_data_sub);
            }
        }
        catch(NullPointerException e){
            System.out.println("Exception occured for sub class "+e.getMessage());
        }
    }
}
public static int createGroup(String property_title) throws SQLException{
```

```
        String[] save_data = {"0",property_title};
        int property_id = database.query(table_property_group, fields_property_group, save_data);
        return property_id;
    }
}
```

Listing 7: "classes/Utils.java"

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package classes;

import edu.smu.tspell.wordnet.NounSynset;
import edu.smu.tspell.wordnet.Synset;
import edu.smu.tspell.wordnet.SynsetType;
import edu.smu.tspell.wordnet.WordNetDatabase;
import java.net.URL;

/**
 *
 * @author Arooj
 */
public class Utils {
    public static boolean validateHTTP_URI(String uri) {
        final URL url;
        try {
            url = new URL(uri);
        } catch (Exception e1) {
            return false;
        }
        return "http".equals(url.getProtocol());
    }
    public static boolean isValidWord(String word){
        System.setProperty("wordnet.database.dir", "E:/WordNet/2.1/dict/");
        WordNetDatabase database = WordNetDatabase.getFileInstance();
        NounSynset nounSynset;
        Synset[] synsets = database.getSynsets(word, SynsetType.NOUN);
        return synsets.length > 0;
    }
}
```

Listing 8: "ontology_processor/OntologyParser.java"

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ontology_processor;

import classes.Classes;
import classes.Inflector;
import classes.NameSpaces;
import classes.Properties;
import classes.Utils;
import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.util.iterator.ExtendedIterator;
import config.config;
import config.database;
import java.sql.SQLException;

/**
 *
```

```
 * @author Arooj Fatima
 */
public class OntologyParser {
    /**
     * The parse() functions parses namespaces 1 at a time from String ONT_URL
     * and fetch classes and properties.
     * The fetched classes and properties are dumped into the database.
     * @param ONT_URL
     * @param ns_short
     * @throws SQLException
    */
    private final static String table_class_map = "class_map";
    private final static String[] fields_class_map = {"id", "class", "term"};

    public static void parse(String ONT_URL, String ns_short) throws SQLException{
        int domain_id = config.domain_id;
        String domain_short = config.domain_short;

        OntModel model = ModelFactory.createOntologyModel();
        model.setStrictMode(false);
        model.read(ONT_URL);
        //NameSpaces.saveNameSpaces(model, domain_id); // don't need it if we are parsing directly from RDF
        // set current class null
        OntClass thisClass = null;
        // find list of classes from the namespace model
        ExtendedIterator classes = model.listClasses();
        int i = 0;
        while (classes.hasNext())
        {
            try{
                thisClass = (OntClass) classes.next();
                if(thisClass.getLocalName() != null && !(thisClass.getLocalName().isEmpty())){ // don't save if class
                     title is empty
                    System.out.println("Class Found :  "+thisClass.toString()+"--"+thisClass.getLocalName() );
                    if(Utils.isValidWord(thisClass.getLocalName().toLowerCase())){
                        Inflector inf = new Inflector();
                        String plural = inf.pluralize(thisClass.getLocalName().toLowerCase());
                        System.out.println("Class Found :  "+thisClass.toString()+"--"+plural );

                        String[] save_class_map_singular = {"0", thisClass.toString(),
                             thisClass.getLocalName().toLowerCase()};
                        database.query(table_class_map, fields_class_map, save_class_map_singular);

                        String[] save_class_map_plural = {"0", thisClass.toString(), plural};
                        database.query(table_class_map, fields_class_map, save_class_map_plural);

                    }

                    Classes.saveClass(thisClass, domain_id, domain_short, ns_short);
                    /********************* Get Equivalent Classes ****************************/
                    Classes.setEquivalentClasses(thisClass, ns_short);
                    /********************* Get Sub Classes ****************************/
                    Classes.setSubClasses(thisClass);
                    /********************* Get Super Classes ****************************/
                    Classes.setSuperClasses(thisClass);
                    /********************* Get Class Properties ****************************/
                    Properties.saveProperty(thisClass, domain_id, domain_short);
                }
            }
            catch(com.hp.hpl.jena.ontology.ConversionException e){
                System.out.println("---Exception found OntologyParser.java line 71: "+e.getLocalizedMessage());
            }
        }
     }
}
```

Listing 9: "ontology_processor/InstanceParser.java"

```
/*
 * To change this license header, choose License Headers in Project Properties.
```

```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ontology_processor;

import classes.Classes;
import classes.NameSpaces;
import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.StmtIterator;
import config.*;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Map;

/**
 *
 * @author Arooj Fatima
 */
public class InstanceParser {
    private final static String[] fields = {"id", "term", "property", "property_uri", "rdf_type", "note"};
    private final static String table = "bag_of_words";
    private final static String table_ontologies = "namespaces";
    private final static String[] fields_ontologies = {"id", "short", "namespace", "domain_id"};

    public static int domain_id = config.domain_id;
    public static String domain_short = config.domain_short;

    public static void fetchInstances(String file_url, String type) throws FileNotFoundException, SQLException{
        Model model = ModelFactory.createDefaultModel();
        model.read(new FileInputStream(file_url),null,type) ;
        InstanceParser.processNameSpaces(model); //loads classes and properties
        InstanceParser.saveInstances(model); // loads bag of keywords
    }
    public static void processNameSpaces(Model model) throws SQLException{
        Map<String, String> nsPrefixMap = model.getNsPrefixMap();
        Iterator it = nsPrefixMap.entrySet().iterator();
        ArrayList ns = new ArrayList();
        while (it.hasNext()) {
            Map.Entry pair = (Map.Entry)it.next();
            System.out.println(pair.getKey() + " = " + pair.getValue());
            if(pair.getKey().toString() != null && !(pair.getKey().toString().isEmpty())){
                //ArrayList data = null; //enable if want to run classes etc anyway and comment NameSpaces.saveNameSpace
                ArrayList data = database.getResults(table_ontologies, new String[]{"short"},
                        " namespace=\""+pair.getValue().toString()+"\" AND domain_id='"+domain_id+"' ");
                if(data == null || data.isEmpty()){
                    NameSpaces.saveNameSpace(pair.getValue().toString(), pair.getKey().toString(),
                         InstanceParser.domain_id);
                    ns.add(pair);
                }
            }
            it.remove(); // avoids a ConcurrentModificationException
        }
        //OntologyParser.parse("http://bloody-byte.net/rdf/dc_owl2dl/dcterms.rdf");
        Iterator iterator = ns.iterator();
        while(iterator.hasNext()){
            Map.Entry ns_pair = (Map.Entry)iterator.next();
            OntologyParser.parse(ns_pair.getValue().toString(), ns_pair.getKey().toString());
            iterator.remove(); // avoids a ConcurrentModificationException
        }
    }
    public static void saveInstances(Model model) throws SQLException{
        System.out.println("************************** Save Instances ***********************");
        StmtIterator stmt = model.listStatements();
        int i=0;
```

```
while(stmt.hasNext()){
    String classes = null;
    String[] temp_classes;
    String properties = null;
    com.hp.hpl.jena.rdf.model.Statement stm = stmt.next();
    Resource res = model.getResource(stm.getSubject().toString());
    com.hp.hpl.jena.rdf.model.Statement stmt1      = stmt.nextStatement();
    //Find all properties for that particular resource
    StmtIterator  prop = res.listProperties();
    String class_uri = "";
    String property_uri = "";
    String property_value = "";
    String ns = "";
    String name[] = null;
    int keyword_id = 0, inc =0;
    int k=0;
    while(prop.hasNext()){
        com.hp.hpl.jena.rdf.model.Statement  thisProperty =  prop.next();
        //if property is type - class
        if("type".equals(thisProperty.getPredicate().getLocalName())){
            class_uri = thisProperty.getObject().toString();
            classes += ", "+class_uri;
            System.out.println("Class Data "+class_uri+"---"+thisProperty.getSubject().getLocalName());
        }else{
            //if("name".equals(thisProperty.getPredicate().getLocalName())){
                property_uri = thisProperty.getPredicate().toString();
                property_value = thisProperty.getObject().toString();
                if(property_value.contains("^^")){
                    String[] temp_prop = property_value.split("^^");
                    property_value = temp_prop[0];
                    System.out.println("contains---"+property_value);
                }
                ns = NameSpaces.getNameSpaceShort(thisProperty.getPredicate().getNameSpace(),
                     InstanceParser.domain_id);
                name =  property_value.split("@en");
                RDFNode   object    = thisProperty.getObject();      // get the object
                 System.out.println(">>>>>>>> -------"+thisProperty.getPredicate().getLocalName());
                if (object instanceof Resource) {

                } else {
                    // object is a literal
                    properties += ";;"+property_uri+":_:"+property_value+":_:"
                            + ""+ns+":"+thisProperty.getPredicate().getLocalName();
                }
            }
            System.out.println(class_uri+"---"+property_uri+"---"+property_value);
            //classes[k] = class_uri;
            inc++;
            k++;
            if(inc >100) {
                //break;
            }
        //}
    }
    if(classes!=null && properties!=null){
        classes = classes.substring(5, classes.length()); // remove null;
        //String child = Classes.findChild(classes);
        properties = properties.substring(6, properties.length()); // remove null;;
        System.out.println("classes -> "+classes);
       // System.out.println("Child of classes is ---"+child);
        System.out.println("Properties -> "+properties);
        String[] list = properties.split(";;");
        for (String list1 : list) {
            String[] pname = list1.split(":_:");
            System.out.println("Saving ->>>>>>>>>>>>>> "+pname[1]);
            String[] save_data = {"0",pname[1],pname[2], pname[0],classes, stm.getSubject().toString()};
            database.query(table, fields, save_data);
        }
    }
    i++;
   // if(i > 100) break;

}
```

```
    }
}
```

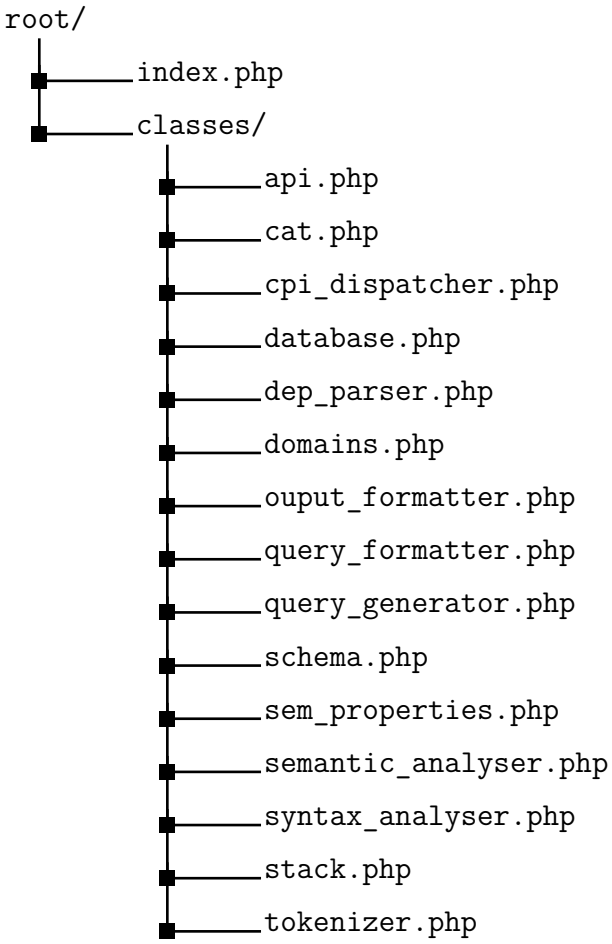
Listing 10: "ontology_processor/RDFParser.java"

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ontology_processor;

import classes.NodePaths;
import java.io.File;
import java.util.ArrayList;
import java.util.HashMap;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

/**
 *
 * @author Arooj Fatima
 */
public class RDFParser {
    private HashMap<String,String> clsList = new HashMap<String,String>();
    private ArrayList<String> paths = new ArrayList<>();
    private String property;
    private String path;
    private final String file;
    private int totalcount = 0;

    public RDFParser(String filepath) {
        this.file=filepath;
    }

    public void readRDF() {
        try {

        File file = new File(this.file);

        DocumentBuilder dBuilder = DocumentBuilderFactory.newInstance()
                                .newDocumentBuilder();
        Document doc = dBuilder.parse(file);
        if (doc.hasChildNodes()) {
            this.parseDocument(doc.getDocumentElement().getChildNodes());
        }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
    public void parseDocument(NodeList nodeList) {
        String tlClass = null;
        //System.out.println(nodeList.getLength());
        for (int count = 0; count < nodeList.getLength(); count++) {
            Node tempNode = nodeList.item(count);
            // make sure it's element node.
            if (tempNode.getNodeType() == Node.ELEMENT_NODE) {
                NodePaths np = new NodePaths(tempNode.getChildNodes());
                np.processNode();
            }
        }
    }
}
```

Appendix II - Implementation of Query Interface

This section provides the code for the implementation of the Framework modules that take part in the process of translating user query to SPARQL query and fetch data from SPARQL endpoints i.e. the Query Optimiser, the Query Formatter, API and the Result Optimiser.

Code Listing of Query Interface (Prototype)



Listing 11: "index.php"

```
<?php
include("autoload.php");
require_once( "libs/sparqllib.php" );
$results = null;
$strs = array();
$strs[0] = "simon books"; //CI
$strs[1] = "list authors of physics"; //CP
$strs[2] = "simon books"; //CI //
$strs[3] = "book authors"; //PC
$strs[4] = "person name"; //PC
$strs[5] = "person book"; //CC
$strs[6] = "Stanislaw Piotowicz"; //II
$strs[7] = "first name and last name"; //PP
$strs[8] = "person name"; //PC related
$strs[9] = "books"; //C
$strs[10] = "authors"; //P
$strs[11] = "simon";//I

$index = isset($_GET['s']) ?  $_GET['s'] : 0;
$domain = isset($_GET['domain']) ?  $_GET['domain'] : 'BL';
$str = isset($strs[$index])? $strs[$index] : $index;
$qf = new Query_formatter($db);
$op_formatter = new Output_formatter();
$qg = new Query_generator($db, $str);
$cached_query = $qf->getCachedQuery($str, $domain);
if($cached_query=="") {
    $sa = new Semantic_analyser($db, $str);
    $schema = implode(",",$qg->schema);
    $results = $qg->results;
}else{
    $q = $cached_query['sparql_query'];
    $schema = $cached_query['display_schema'];
    $results = json_decode($cached_query['results'], true);
}
$url = "?s=".$str;
?>
<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title><?php echo $str? $str.' - ': '';?>SIRF Demo</title>
    <link href="<?php echo HTTP_CSS;?>style.css" type="text/css" rel="stylesheet"/>
</head>
<body>
<header id="header">
    <div class="container">
        <div class="row">
            <a href="/" id="logo"><img src="<?php echo HTTP_IMG;?>logo.png" alt="SIRF" title="Home"></a>
            <form class="search-box" method="GET" action="">
                <div class="wrapper-search">
                    <div class="search-input">
                        <input id="search-input" name="s" type="text" autocomplete="off" placeholder="Quick Search"
                            value="<?php echo @$str;?>">
                    </div>
                    <div class="search-button">
                        <button type="button" id="search-btn" onClick="submitFrom();" name="search-btn"> </button>
                    </div>
                </div>
            </form>
        </div>
    </div>
    <div class="divider"> </div>
</header>
<div class="wrapper container">
        <div class="suggestions">
     <div class="suggestion"><span class="title">You Searched: </span><span class="text"><?php echo $str;
         ?></span></div>
        </div>
    <div class="results">
```

```
<?php
  if($results) {
    $schemaArr = explode(",", $schema);
    $format_desc = false;
    if ($domain == 'BL') {
        $url = $schemaArr[0];
        $title = $schemaArr[1];
        $url2  = !empty($schemaArr[2])?$schemaArr[2] : '';
        $desc  = !empty($schemaArr[3]) ? $schemaArr[3] : '' ;
        $rows = $results['rows'];
    } else if ($domain == 'DBP') {
        $url = $schemaArr[0];
        $title = $schemaArr[1];
        $thumb  = !empty($schemaArr[2])? $schemaArr[2] : '';
        $url2  = !empty($schemaArr[3])?$schemaArr[3] : '';
        $desc  = !empty($schemaArr[4]) ? $schemaArr[4] : '' ;
        if(isset($schemaArr[2]) && strpos($schemaArr[2],'thumbnail') === false && !$desc) {
            $desc = $thumb;
            $thumb = false;
            $format_desc = true;
        }
        $rows = $results['results']['bindings'];
    }
    foreach($rows as $row) { ?>
    <div class="result">
        <?php if($domain == "DBP"){?>
        <?php
        $thumb_url = isset($row[$thumb])? $row[$thumb]['value']:'';
        if($thumb_url){
            $thumb_url = str_replace('?width=300', '?width=100&height=75', $thumb_url);
        ?>
        <div class="thumb"><img src="<?php echo $thumb_url;?>" alt="thumbnail"/></div>
        <?php } } ?>
        <div class="description">
        <div class="title"><a href="<?php echo urldecode($row[$url]['value']); ?>" target="_blank"><?php  echo
             $row[$title]['value']; ?></a></div>
        <div class="link">
            <?php if($url2){ ?>
            <a href="<?php echo urldecode($row[$url2]['value']); ?>" target="_blank"><?php echo
                urldecode($row[$url2]['value']); ?></a>
            <?php } else {?>
            <a href="<?php echo urldecode($row[$url]['value']); ?>" target="_blank"><?php echo
                urldecode($row[$url]['value']); ?></a>
            <?php } ?>
        </div>
        <div class="detail">
            <?php
            if($format_desc){
               echo $op_formatter->getOutput($row[$desc], $desc);
            } else
            echo $desc? $row[$desc]['value']: '';
            ?>
        </div>
        </div>
    </div>
    <?php
    }
    } else {
        echo "Your search - <strong>".$str."</strong> - did not match any documents";
    }
?>
</div>
<div class="domains">
    <div class="wrapper-demo active">
        <div id="dd" class="wrapper-dropdown-4 active"><span class="dd-selected">See also ...</span>
            <ul class="dropdown">
                <li class="<?php echo $domain == 'BL'? 'active': '';?>"><label for="el-1"><a href="<?php echo $url
                    = "?s=".$str."&domain=BL";?>">British Library</a></label></li>
                <li class="<?php echo $domain == 'DBP'? 'active': '';?>"><label  for="el-2"><a href="<?php echo
                    $url = "?s=".$str."&domain=DBP";?>">DBPedia</a></label></li>
            </ul>
        </div>
    </div>
```

```
    </div>
</div>
<footer>
</footer>
<script src="<?php echo HTTP_JS;?>jquery-1.10.2.min.js"></script>
<script src="<?php echo HTTP_JS;?>jquery-ui-1.10.4.custom.min.js"></script>
<script>
var href = '/';
function submitFrom() {
    var query =$("#search-input").val();
    if(query)
        window.location = href+"?s="+query;
}
$(function() {
    var cache = {};
    $( "#search-input" ).autocomplete({
        minLength: 2,
        select: function( event, ui ) {
            item = ui.item;
            $("#search-input").val(item.label);
            $("#selected").val("1");
            $("#search-btn").click();
        },
        source: function( request, response ) {
            var term = request.term;
            $.getJSON( "search.php", request, function( data, status, xhr ) {
                response( data );
            });
        }
    });
    $(".suggestion .text").click(function(){
        $("#search-input").val($(this).html());
        $("#selected").val("1");
        $("#search-btn").click();
    });
});
</script>
</body>
</html>
```

Listing 12: "classes/api.php"

```
<?php
class API {
 private $endpoint;//sparql endpoint
 private $results;
 public function __construct($endpoint){
        $this->endpoint = $endpoint;
 }
 public function getResults($query) {
        sparql_connect($this->endpoint);
  $results = sparql_query($query);
  $this->results = $results;
  return $results;
 }
}
?>
```

Listing 13: "classes/cat.php"

```
<?php
class Cat {
 private $db;//sparql endpoint
    private $object_type = 2; // type 2 = object type
    private $data_type = 1; // type 1 = data type
    private $property_types = array();
    private $missing_links;
 public function __construct($db){
```

```
        $this->db = $db;
        $this->missing_links = new Missing_links();
 }
    private function getPropertyType($property, $short = "") {
        if(isset($this->property_types[$short])) {
            return $this->property_types[$short];
        }

        $sql = "SELECT * properties WHERE property ='$property'";
        $query = $this->db->query($sql);
        $row = $query->row;
        if($row){
            $this->property_types[$row['property_short']] = $row['property_type'];
            return $row['property_type'];
        } else {
            return 4; //property tupe not found;
        }

    }
    public function calculateCAT($cnd_concepts, $dcl_concepts= ""){
        $missinglinks = $this->missing_links->getMissingLinks($cnd_concepts);
        $cat = array();
        if(!empty($missinglinks)) {
            foreach($cnd_concepts as $row){
                if($row['stype'] == 'P'){ //P = property
                    if($row['type'] == $this->object_type && !empty($row['range'])){
                        $range = $this->missing_links->getRange($row);
                        $cat['class'] = $range;
                    }
                    else if ($row['type'] == $this->object_type && !empty ($row['value'])) {
                        $cat['class'] = $row['class'];
                    }
                }
            }
        } else {
            $property = array();
            $isclass = false;
            $class = "";
            foreach($cnd_concepts as $row){
                if($row['type'] == 'CI'){ //Class Instance Relation
                    $class = $row['class'];
                    $isclass = true;
                } elseif($row['type'] == "PI"){
                    if($row['property']['type'] != $this->data_type){ //object type property
                        $class = $row['property']['range'];
                        $isclass = true;
                    } else { //if property type is data type
                        $class = $row['property']['class'];
                        $property[] = $row['property']['property'];
                    }
                } elseif($row['type'] == "PC"){
                    if($row['property']['type'] != $this->data_type){ //object type property
                       $class = $row['property']['range'];
                       $isclass = true;
                    } else{ //if property type is data type
                       $class = $row['property']['class'];
                       $property[] = $row['property']['property'];
                    }
                }
            }
            if(!empty($class)){
                $cat['class'] = $class;
            } else if(empty($class) && !empty ($property)){
                array_unique($property);
                $cat['property'] = $property;
            }
        }

        return array('isclass'=>$isclass, 'cats'=>$cat);
    }
}
```

Listing 14: "classes/cpi_dispatcher.php"

```
<?php
class Cpi_dispatcher
{
    private $db;
    public $domains = array();
    public $concepts = array();
    public $domain = 0;
    public $cpi_relations = array();
    public function __construct($db)
    {
        $this->db = $db;
    }

    public function getRelations() {
        if(count($this->cpi_relations) > 0){
            return $this->cpi_relations;
        } else {
            return false;
        }
    }
    public function findRelations($concept1, $concept2, $equal = false){
        if($equal) { // if only one word
            $concept = $concept1;
            if($concept2)
                $concept = $concept2;
            if($concept["type"]=="P"){
               if($concept['domain'] != "" && $concept['range'] != ""){
                   $this->cpi_relations[] = array("type"=>"P",
                    "property"=>$concept["property"], "class"=>$concept, "relation"=>array());
               } else {
                   $relation = $this->indirectPath("", $concept["property"], "", "P");
                    if($relation <> null){
                        $this->cpi_relations[] = array("type"=>"P", "property"=>$concept["property"],
                             "class"=>$concept, "relation"=>$relation);
                    } else{
                        $this->cpi_relations[] = array("type"=>"P",
                            "property"=>$concept["property"], "class"=>$concept, "relation"=>array());
                    }
               }
            }
            else if($concept["type"]=="C"){
               $this->cpi_relations[] = array("type"=>"C","class"=>$concept, "relation"=>array());
            }
            else if($concept["type"]=="I"){
               $this->cpi_relations[] = array("type"=>"I","class"=>$concept['class'], "instance"=>$concept,
                    "relation"=>array());
            }
        } else {
            if($concept1["type"]=="I" && $concept2["type"]=="I"){
                $this->findInstanceInstanceRelation($concept1, $concept2, "II");
            }
            else if(($concept1["type"]=="I" && $concept2["type"]=="C") || $concept1["type"]=="C" &&
                 $concept2["type"]=="I"){
               $this->findClassInstanceRelation($concept1, $concept2, "CI");
            }
            else if(($concept1["type"]=="I" && $concept2["type"]=="P") || ($concept1["type"]=="P" &&
                 $concept2["type"]=="I")){
                $this->findPropertyInstanceRelation($concept1, $concept2, "PI");
            }
            else if($concept1["type"]=="C" && $concept2["type"]=="C"){
                $this->findClassClassRelation($concept1, $concept2, "CC");
            }
            else if(($concept1["type"]=="C" && $concept2["type"]=="P") || ($concept1["type"]=="P" &&
                 $concept2["type"]=="C")){
                $this->findPropertyClassRelation($concept1, $concept2, "CP");
            }
            else if($concept1["type"]=="P" && $concept2["type"]=="P"){
                $this->findPropertyClassRelation($concept1, $concept2, "PP");
            }
```

```
    }
}
public function findClassClassRelation($class1, $class2) {
    $properties = $this->getPropertyByDomainOrRange($class1['class'], $class2['class']);
    if($properties){
        echo "<br/>Inside CC, domain range properties found.<br/>";
    }else {
        $relation = $this->indirectPath($class1["class"], $class2["class"], "", "CC");
        if($relation <> null){
            $this->cpi_relations[] = array("type"=>"CC", "class"=>$class1["class"], "class"=>$class2,
                 "relation"=>$relation);
        }
    }

}

public function findInstanceInstanceRelation($instance1, $instance2) {
    if($instance1['class'] == $instance2['class']){
        $this->cpi_relations[] = array("type"=>"II", "class"=>$instance1["class"],
             "instance"=>$instance2,"relation"=>array());
    }else {
        $properties = $this->getPropertyByDomainOrRange($instance1['class'], $instance2['class']);
        if($properties){
            echo "<br/>Inside CC, domain range properties found.<br/>";
        }else {
            $relation = $this->indirectPath($instance1["class"], $instance2["class"], "", "II");
            if($relation <> null){
                $this->cpi_relations[] = array("type"=>"II", "class"=>$instance1["class"], "instance"=>$instance2,
                     "relation"=>$relation);
            }
        }
    }

}

public function findClassInstanceRelation($concept1, $concept2) {
    if($concept1["type"]=="I") {
        $instance =$concept1;   $class = $concept2;
    }
    else{
        $instance =$concept2;    $class = $concept1;
    }
    if($class['class'] == $instance['class']){ // class of instance equal to class
        echo "Related CC-> IC";//related

    }else {
        $properties = $this->findPropertyByClassAndInstance($class, $instance);
        if($properties){

        }else{ //find indirect path
            $relation = $this->indirectPath($class["class"], "", $instance["class"], "CI");
            if($relation <> null){
                $this->cpi_relations[] = array("type"=>"CI", "class"=>$class["class"], "instance"=>$instance,
                     "relation"=>$relation);
            }
        }
    }

}
public function findPropertyClassRelation($concept1, $concept2){
    if($concept1["type"]=="C") {
        $class =$concept1;   $property = $concept2;
    }
    else{
        $class =$concept2;    $property = $concept1;
    }

    if($property["property_type"]==0){ // is a data type property
        if(!empty($property['class'])){
            if($class["class"] == $property["class"]){
                $this->addRelatedPaths($class, "", $property, "CP");
            }
        }else if($class["class"] == $property["domain"]){
```

```
            $this->addRelatedPaths($class, "", $property, "CP");
        } else {
            $relation = $this->indirectPath($class["class"], $property["property"],"", "CP");
            if($relation <> null){
                $this->cpi_relations[] = array("type"=>"CP", "property"=>$property["property"], "class"=>$class,
                    "relation"=>$relation);
            }
        }
    }else  if($property["property_type"]==1 || $property["property_type"]==3){ // is an object type property with a
         domain and range or unknown
        if($class["class"] == $property["range"] || $class["class"] == $property["domain"]){
            $this->addRelatedPaths($class, "", $property, "CP");
        }
        else{
            $relation = $this->indirectPath($class["class"], $property["property"],"", "CP");
            if($relation <> null){
                $this->cpi_relations[] = array("type"=>"CP", "property"=>$property["property"], "class"=>$class,
                    "relation"=>$relation);
            }
        }
    }
}

public function findPropertyInstanceRelation($concept1, $concept2){
    if($concept1["type"]=="I") {
        $instance =$concept1;   $property = $concept2;
    }
    else{
        $instance =$concept2;    $property = $concept1;
    }

    if($property["property_type"]==0){ // is a data type property
        if($instance["property"] == $property["property"]){
            // related
        }
    }else  if($property["property_type"]==1 || $property["property_type"]==3){ // is an object type property with a
         domain and range or unknown
        if($instance["class"] == $property["range"] || $instance["class"] == $property["domain"]){
            // related
        }
        else{
            $relation = $this->indirectPath("", $property["property"], $instance["class"], "PI");
            if($relation <> null){
                $this->cpi_relations[] = array("type"=>"PI", "property"=>$property["property"],
                    "instance"=>$instance, "relation"=>$relation);
            }
        }
    }
}
public function indirectPath($class="", $property="", $instance ="", $type = ""){
    $relations = null;
    if($type == "PI") {
       $sql = "SELECT *
 FROM indirect_paths
 WHERE ((first_node='".$this->db->escape($instance)."'
   AND property_uri='".$this->db->escape($property)."')
  OR (last_node='".$this->db->escape($instance)."'
   AND property_uri='".$this->db->escape($property)."')
                   AND domain_id = ".$this->db->escape($this->domain).")";

    } else if($type =="CI"){
        $sql = "SELECT *
 FROM indirect_paths
 WHERE ((first_node='".$this->db->escape($class)."'
   AND last_node='".$this->db->escape($instance)."')
  OR (last_node='".$this->db->escape($class)."'
   AND first_node='".$this->db->escape($instance)."')
                   AND domain_id = ".$this->db->escape($this->domain).")";
    } else if($type == "CP"){
        $sql = "SELECT *
 FROM indirect_paths
 WHERE ((first_node='".$this->db->escape($class)."'
   AND property_uri='".$this->db->escape($property)."')
```

```
  OR (last_node='".$this->db->escape($class)."'
   AND property_uri='".$this->db->escape($property)."')
                    AND domain_id = ".$this->db->escape($this->domain).")";
    }else if($type == "CC" || $type == "II"){
        $sql = "SELECT *
 FROM indirect_paths
 WHERE ((first_node='".$this->db->escape($class)."'
   AND last_node='".$this->db->escape($property)."')
  OR (last_node='".$this->db->escape($class)."'
   AND first_node='".$this->db->escape($property)."')
                    AND domain_id = ".$this->db->escape($this->domain).")";
    } else if($type == "P"){
        $sql = "SELECT *
 FROM indirect_paths
 WHERE property_uri = '".$this->db->escape($property)."'
                    AND domain_id = ".$this->db->escape($this->domain)."";
    } else {

        return false;
    }

    $query = $this->db->query($sql);
    if($query->num_rows > 0){
        foreach($query->rows as $row) {
            $relation = array();
            $domain = $row["first_node"];
            $range = $row["last_node"];
            $relation["rel_property"] = $row['property_uri'];
            $relation["class"] = $class;
            $relation["domain"] = $domain;
            $relation["range"] = $range;
            $relation["links"] = $row["links"];
            $relation["rel_property_name"] = $row['property'];
            $relations[] = $relation;
        }
    }
    return $relations;
}
public function findPropertyByClassAndInstance ($class, $instance){
    $sql = "SELECT * FROM `properties` WHERE "
            . "`domain` LIKE '".$class['class']."' and `property_range` LIKE '".$instance['class']."'"
            . " AND "
            . "`domain` LIKE '".$instance['class']."' and `property_range` LIKE '".$class['class']."'";
    $query = $this->db->query($sql);
    if($query->num_rows){
        return $query->rows;
    } else {
        return false;
    }
}

public function getPropertyByDomainOrRange($domain, $range) {
    $sql = "SELECT * FROM `properties` WHERE "
            . "`domain` LIKE '".$domain."' and `property_range` LIKE '".$range."'"
            . " AND "
            . "`domain` LIKE '".$range."' and `property_range` LIKE '".$domain."'";
    $query = $this->db->query($sql);
    if($query->num_rows){
        return $query->rows;
    } else {
        return false;
    }
}
/*
 * This function will be used to add relations when given phrases are related.
 */
public function addRelatedPaths($class = false, $instance = false, $property=false, $type ="") {
    if($type == "CP") {
        $row_relation = $this->getPropertyByCP(array('class'=>$class['class'], 'property'=>$property['property']),
             $type);
        $relation = array('class'=>$row_relation['class'], 'domain'=>$row_relation['domain'],
            'range'=>$row_relation['property_range'],'links'=>'{'.$row_relation['class'].",".
            $row_relation['property_short'].",".$row_relation['class'].'}');
```

```
            $this->cpi_relations[] = array("type"=>$type, "property"=>$property["property"], "class"=>$class,
                "relation"=>array($relation));
        }
    }
    public function getPropertyByCP($data, $type = ""){
        echo $sql = "SELECT * FROM properties WHERE property='".$data['property']."' AND class='".$data['class']."'";
        $query = $this->db->query($sql);
        return $query->row;
    }
}
```

Listing 15: "classes/database.php"

```
<?php
final class Database {
 private $link;

 public function __construct($hostname, $username, $password, $database) {
  $this->link = new \mysqli($hostname, $username, $password, $database);

  if ($this->link->connect_error) {
   trigger_error('Error: Could not make a database link (' . $this->link->connect_errno . ') ' .
        $this->link->connect_error);
  }
  $this->link->set_charset("utf8");
  $this->link->query("SET SQL_MODE = ''");
 }

 public function query($sql) {
  $query = $this->link->query($sql);

  if (!$this->link->errno) {
   if ($query instanceof \mysqli_result) {
    $data = array();

    while ($row = $query->fetch_assoc()) {
     $data[] = $row;
    }
    $result = new \stdClass();
    $result->num_rows = $query->num_rows;
    $result->row = isset($data[0]) ? $data[0] : array();
    $result->rows = $data;

    $query->close();

    return $result;
   } else {
    return true;
   }
  } else {
   trigger_error('Error: ' . $this->link->error  . '<br />Error No: ' . $this->link->errno . '<br />' . $sql);
  }
 }

 public function escape($value) {
  return $this->link->real_escape_string($value);
 }

 public function countAffected() {
  return $this->link->affected_rows;
 }

 public function getLastId() {
  return $this->link->insert_id;
 }

 public function __destruct() {
  $this->link->close();
 }
}
```

Listing 16: "classes/dep_parser.php"

```
<?php
class Dep_Parser{
 private $data = array();
 private $stack = array();
    private $output = array();
 private $removeatstepone = array('det', 'cop');
    private $questions = array('find', 'fist', 'what', 'how', 'when', 'who', 'list');
 private $ignoreconditions = array('in', 'of', 'with');
    private $nsubjindexes = array();
    private $nmodes = array('nmod:of', 'nmod:for', 'nmod:with', 'nmod:in', 'nmod:by');
    private $nmodesArray = array();
    private $tempArr = array(); //used to saved words index
    private $rootIndex = 0;

 public function __construct($data){
  $this->data = $data;
        $this->stepOne(); //remove unused
        $this->stepThree(); //apply amods
        $this->stepFour(); //apply compounds
        $this->stepFive(); //apply conditions
        $this->stepSix(); //apply ref
        $this->stepSeven(); //apply conjugation
        $this->stepTwo(); //find root should be at the end
 }

    private function getStack($index) {
        if(isset($this->stack[$index])){
            return $this->stack[$index];
        } else {
            return false;
        }
    }

    private function getOutputByIndex($index) {
        if(isset($this->output[$index])){
            return $this->output[$index];
        } else {
            return false;
        }
    }

    private function getValueByIndex($arr, $index, $col){
        if(isset($arr[$index])){
            return isset($arr[$index][$col])? $arr[$index][$col]: false;
        } else {
            return false;
        }
    }
    private function removeOutput($index) {
        if(isset($this->output[$index])){
            unset($this->output[$index]);
        }
    }


    private function removeStack($index) {
        if(isset($this->stack[$index])){
            unset($this->stack[$index]);
        }
    }

    private function removeData($index) {
        if(isset($this->data[$index])){
            unset($this->data[$index]);
        }
    }

    private function removeNsubj() {
        foreach($this->nsubjindexes as $index) {
```

```
        unset($this->nsubjindexes[$index]);
        $this->removeStack($index);
    }
}
/*
 * remove copula and det
 */
public function stepOne() {
    $len = sizeof($this->data);
    for($i=0; $i<$len; $i++) {
        $set = $this->data[$i];
        if(!in_array($set['type'],$this->removeatstepone)){
            $this->stack[$set[1]['index']] = $set;
        }

        if($set['type'] == 'nsubj'){
            $this->nsubjindexes[$set[1]['index']] = $set[1]['index'];
        }
        if(in_array($set['type'], $this->nmodes)) {
           $this->nmodesArray[$set[1]['index']] = array('type'=>$set['type'], '0'=>$set[0]['feature'],
                1=>$set[1]['feature']);
        }
        if($set['type'] == 'root')  $this->rootIndex = $set[1]['index'];
        $this->tempArr[$set[1]['index']] = $set[1]['feature'];
    }

}

/*
 * second step find root Category
 */
public function stepTwo(){
   $rIndex = $this->rootIndex;
   $root = $this->stack[$rIndex];//root element
   if(in_array(strtolower($root[1]['feature']), $this->questions)){ //if question keep root in stack
        $rcIndex = $this->findRootIndex($root[1]['feature'], $this->stack);
        if($rcIndex) {
            $rootCat = $this->stack[$rcIndex];
            $this->output[$root[1]['index']] = array('type'=>'root', $rootCat[0]['feature'].' '.
                 $rootCat[1]['feature']);
            $this->removeStack($rcIndex);
        } else {
            $this->output[$root[1]['index']] = array($root[1]['feature']);
        }
   }
   if(!$this->output){
        if($root['type'] == 'root'){
            $this->output[$root[1]['index']] = array($root[1]['feature']);
        }
   } else {
        $this->removeStack($rIndex);//remove root not needed
   }
}

/*
 * third step hand amod
 */
public  function stepThree() {
    foreach($this->stack as $i=>$set) {
        if($set['type'] == 'amod'){
            //$this->stack[$set[0]['index']] = $this->leftArc($set, true);
            $this->output[$set[0]['index']] = $this->leftArc($set, true);
            $this->removeStack($i);//enable this later to if seen issue
        }
    }
    return array('data'=>$this->data, 'stack'=>$this->stack);
}

/*
 * fourth Step handle compounds
 */
public function stepFour() {
    foreach($this->stack as $i=>$set) {
```

```
        if($set['type'] == 'compound'){
            if(isset($this->output[$set[0]['index']])) {
               $this->output[$set[0]['index']] = array_replace($this->output[$set[0]['index']],
                    $this->leftArc($set, true));
            } else {
                $this->output[$set[0]['index']] = $this->leftArc($set, true);
            }
            $this->removeStack($i);
        }
    }
}

/*
 * fivth step, handle cases
 */
public function stepFive(){
    foreach($this->stack as $i=>$set) {
        if($set['type'] == 'case'){
            $ofindex = $set[1]['index']-1;
            if($set)
            $arr2 = array($set[0]['index']=>$set[0]['feature']);
             if(isset($this->output[$set[0]['index']])) {
                 ksort($this->output[$set[0]['index']]);
                 $arr2 = array(implode(' ', $this->output[$set[0]['index']]));
             }
             if(isset($this->output[$ofindex]) && !isset($this->output[$ofindex]['case'])){
                 $feature = $this->data[$ofindex][1]['feature'];
                 $findkey = $this->findKeyValue($feature, $this->output, false);
                 if($findkey != $ofindex) {
                     $arr1 = $this->output[$ofindex];
                     $this->removeOutput($ofindex);
                 } else {
                     $arr1 = array($feature);
                 }

             }else {
                $ofStack = $this->getStack($ofindex);
                if($ofStack) {
                    $arr1 = array($this->getValueByIndex($ofStack, 1, 'feature'));
                    if(strstr('cc', $this->stack[$ofindex]['type']) !== false)
                        $this->removeStack($ofindex);
                } else if(isset($this->tempArr[$ofindex])){
                    $arr1 = array($this->tempArr[$ofindex]);
                } else {
                    $arr1 = array($this->data[$ofindex][1]['feature']);
                }

             }

ksort($arr1);
            ksort($arr2);
            $arr1 = implode(' ', $arr1);
            $arr2 = implode(' ', $arr2);
$this->output[$set[0]['index']] = array($ofindex=>$arr1, 'case'=>$set[1]['feature'], $set[0]['index']=>$arr2);
            $this->removeStack($i);
        }
        if(in_array($set['type'], $this->nmodes)){
            $this->removeStack($i);
        }
    }
    //print_r($this->stack);

}

/*
 * sixth Step, handle references
 */
 public function stepSix(){
    foreach($this->stack as $i=>$set) {
        if($set['type'] == 'ref'){
            //$this->output[$set[1]['index']] = $this->leftArc($set, true);
            $this->removeStack($i);
        } else if($set['type'] == 'acl:relcl') {
```

```
            $this->output[$set[1]['index']] = $this->leftArc($set, true);
            $this->removeStack($i);
        }
    }
  }

  /*
   * seventh Step, handle conjugations
   */
   public function stepSeven(){
      foreach($this->stack as $i=>$set) {
          if(isset($set['type']) && $set['type'] == 'conj:and'){
              if($set[0]['feature'] != $set[1]['feature']){
                 $id1 = $this->findKeyValue($set[0]['feature'], $this->output);
                 $id2 = $this->findKeyValue($set[1]['feature'], $this->output);
              } else {
                 $id1 = isset($this->output[$set[0]['index']]) ? $set[0]['index'] :
                      $this->findKeyValue($set[0]['feature'],
                          $this->output);
                 $id2 = isset($this->output[$set[1]['index']]) ? $set[1]['index'] :
                      $this->findKeyValue($set[1]['feature'],
                          $this->output);
              }
              //echo $id1."".$id2;
              $arr1 = $this->output[$id1];
              $arr2 = $this->output[$id2];
              $this->removeOutput($id1);
              $this->removeOutput($id2);
              $this->output[$id2] = array('type' => 'and', implode(' ', $arr1), implode(' ',$arr2));
              $this->removeStack($i);
          } else if(isset($set['type']) && $set['type'] == 'cc') {
              $this->removeStack($i);
          }
      }
  }

  /*
*Following relation belongs to left arc
*/
public function leftArc($set,$arr =true) { //left arc
 $return = $set[1]['feature'].' '.$set[0]['feature'];
 if ($arr) {
          $return = array($set[0]['index'] => $set[0]['feature'], $set[1]['index'] => $set[1]['feature']);
          ksort($return);
      }
      return $return;
  }
public function findKeyValue($key, $data = array(), $debug = false) {
 if($debug) {
  echo "<br>====================";
  echo "key=".$key."<br>";
  print_r($data);
 }
 $return = '';
 foreach($data as $k=>$item) {
  if(is_array($item)){

              $find = array_search($key, $item);
              //$find = $this->findKeyValue($key, $item, $debug);
   if($debug) { echo $find; print_r($item);}
   if($find !== false){$return = $k;}
  } else if($item == $key) {
              $return = $k;
          }
 }

 if($debug){
  echo "Return = $return, <br>====================<br>";
 }
 return $return;
}

public function findRootIndex($key, $data = array()) {
```

```
  $return = '';
        $key = strtolower($key);
  foreach($data as $k=>$item) {
   if(is_array($item)){
                $find = $this->findRootIndex($key, $item);
    if($find !== false && $k > 1){ return $k;}
   } else if(strtolower($item) == $key) {
                $return = $k;
            }
  }
  return $return;
 }

    public function outputStack() {
        echo "Output Stack<br>";
        print_r($this->stack);
    }
    public function getOuput() {
        ksort($this->output);
        if($this->stack) {
            $this->mergeStackWithOutput();
            $this->processStack();
            $this->output = array_replace($this->stack, $this->output);
        }
        return $this->output;
 }

    public function processStack() {
        foreach($this->stack as $i=>$set) {
            $this->stack[$i] = array('type'=>$set['type'], $set[0]['feature'],$set[1]['feature']);
        }
    }

    public function processArray($arr) {
        foreach($arr as $i=>$set) {
            $arr[$i] = array($set[0]['feature'].' ',$set[1]['feature']);
        }
        return $arr;
    }

    public function mergeStackWithOutput() {
        foreach($this->stack as $i=>$set) {
            $output = $this->getOutputByIndex($i);
            if($output && $set['type'] == 'dobj') {
                ksort($output);
                $outputstr = implode(" ", $output);
                $part1 = $part2 = "";
                if($set[0]['index'] == $i) {
                    $part1 = $outputstr;
                } else {
                    $part1 = $set[0]['feature'];
                }
                if($set[1]['index'] == $i) {
                    $part2 = $outputstr;
                } else {
                    $part2 = $set[1]['feature'];
                }

                $newset = array('type'=>$set['type'], $part1, $part2);
                $this->removeStack($i);
                $this->output[$i] = $newset;
            }
        }
    }

}
```

Listing 17: "classes/domains.php"

```
<?php
```

```
class Domains{
    private $db;
    public  $domains =  array();
    public  $concepts =  array();

    public function __construct($db){
        $this->db= $db;
    }
    public  function getMappedDomains($concepts=array()){
  if(isset($concepts["classes"])){
   $this->mapClasses($concepts["classes"]);
  }
  if(isset($concepts["properties"])){
   $this->mapProperties($concepts["properties"]);
  }
        if(isset($concepts["instances"])){
            $this->mapInstances($concepts["instances"]);
        }
        if(sizeof($this->concepts) > 1){
            $intersect = call_user_func_array('array_intersect',$this->concepts);
            return array_unique($intersect);
        } else {
            return $this->concepts;
        }


 }
    /*
     * This function find namesspaces for given concepts and sort them on the basis of usage
    */
    public function mapClasses($classes){
        foreach($classes as $class){
            if(is_array($class["class"])){
                $class_array = array();
                foreach($class["class"] as $c){
                    $class_array[] =  " class='".$this->db->escape(trim($c))."' ";
                }
                $class_array = array_unique($class_array);
                $class_query = implode(" OR ", $class_array);
            }
            else{
                $class_query = "class='".$this->db->escape($class["class"])."'";
            }
            $sql = "SELECT c.id AS class_id, c.`class`, c.`title`, c.`schema`, cd.domain_id
                        FROM classes c
                        LEFT JOIN class_domains cd ON c.id = cd.class_id
                        WHERE c.group_id IN (SELECT group_id FROM classes WHERE ".$class_query.") AND c.`schema` <>
                             'sem'";
            $query = $this->db->query($sql);
            if($query->num_rows > 0){
                foreach($query->rows as $row){
                    $this->domains[$row["domain_id"]]["words"][$class["word"]][$row['class_id']] =
                            array("type"=>$class["type"], "word"=>$class["word"], "class"=>$row["class"]);
                    $this->concepts[$row["class"]][] = $row["domain_id"];
                }
            }
        }
    }
    public function mapProperties($properties)
    {
        foreach($properties as $property){
            $property_query = "property='".$this->db->escape($property["property"])."'";
            $sql = "SELECT p.id AS property_id, p.`property`, p.property_type, p.domain, p.property_range,
                 pd.domain_id, p.class
       FROM properties p
       LEFT JOIN property_domains pd ON p.id = pd.property_id
       WHERE p.property_group IN (SELECT property_group FROM properties WHERE ".$property_query.")
                           AND p.`property_short` NOT LIKE  'sem:%'";
            $query = $this->db->query($sql);
            if($query->num_rows > 0){
                foreach($query->rows as $row){
                    $this->domains[$row["domain_id"]]["words"][$property["word"]][$row['property_id']] =
```

```
                            array("type"=>$property["type"], "word"=>$property["word"], "property"=>$row["property"],
                                "property_type"=>$row["property_type"],"class"=>$row['class'], "domain"=>$row["domain"],
                                "range"=>$row["property_range"]);
                        $this->concepts[$row["property"]][] = $row["domain_id"];
                    }
                }
            }
        }
        public function mapInstances($instances)
        {
            //pr($instances);
            foreach($instances as $instance){
                if(is_array($instance["class"])){
                    $class_array = array();
                    foreach($instance["class"] as $c){
                        $class_array[] =  " class='".$this->db->escape(trim($c))."' ";
                    }
                    $class_array = array_unique($class_array);
                    $class_query = implode(" OR ", $class_array);
                }
                else{
                    $class_query = "class='".$this->db->escape($instance["class"])."'";
                }
               $sql = "SELECT c.id AS class_id, c.`class`, c.`title`, c.`schema`, cd.domain_id
                            FROM classes c
                            LEFT JOIN class_domains cd ON c.id = cd.class_id
                            WHERE c.group_id IN (SELECT group_id FROM classes WHERE ".$class_query.") AND c.`schema` <>
                                 'sem'
                                AND cd.domain_id IS NOT NULL";
                $query = $this->db->query($sql);

                $property_query = "property='".$this->db->escape($instance["property"])."'";
                $sqlp = "SELECT p.id AS property_id, p.`property`, pd.domain_id
           FROM properties p
           LEFT JOIN property_domains pd ON p.id = pd.property_id
           WHERE p.property_group IN (SELECT property_group FROM properties WHERE ".$property_query.") AND
                                 p.`property_short` NOT LIKE  'sem:%'";
          $queryp = $this->db->query($sqlp);
                if($query->num_rows > 0){
                    foreach($query->rows as $row){
                        $this->concepts[$row["class"]][] = $row["domain_id"];
                        if($queryp->num_rows > 0){
                            foreach($queryp->rows as $rowp){
                                $cpid = $row['class_id'].'-'.$rowp["property_id"];
                                $this->domains[$row["domain_id"]]["words"][$instance["word"]][$cpid] =
                                        array("type"=>$instance["type"], "word"=>$instance["word"],"class"=>$row["class"],
                                            "property"=>$rowp["property"]);
                                $this->concepts[$rowp["property"]][] = $rowp["domain_id"];
                            }
                        }
                    }
                }
            }
        }
        public function getEndpoint($domain_id)
        {
            $sqlp = "SELECT endpoint FROM domains WHERE id = " . (int)$domain_id;
            $query = $this->db->query($sqlp);
            if ($query->num_rows > 0) {
                return $query->rows[0]["endpoint"];
            }
            else{
                die("Domain endpoint not found");
            }
        }
        public function filterDomains() {
            $domains =$this->domains;
            $filter = array();
        }
}
```

Listing 18: "classes/output_formatter.php"

```
<?php

/**
 * Description of output_formatter
 *
 * @author Arooj
 */
class Output_formatter {
    //put your code here
    private $string="";
    private $input = array();
    private $key = "";


    public function getOutput($data=array(), $key){
       $this->input = $data;
       $this->key = $key;
       $this->format();
       return $this->string;
    }

    private function format(){
        if(!isset($this->input['datatype'])){
            $this->string = $this->input['value'];
            return;
        }

        list($prefix, $type) = explode("#", $this->input['datatype']);

        list($c, $p) = explode("_", $this->key);
        switch ($type){
            case "integer":
                $this->string = $p." ".number_format($this->input['value'],0);
        }
    }
}
```

Listing 19: "classes/query_formatter.php"

```
<?php
class Query_formatter
{
  private $db;
 public function __construct($db)
 {
  $this->db = $db;
 }

 public function getCachedQuery($str, $domain)
 {
  $sql = "SELECT * FROM sparql_cache WHERE user_query = '".$str."' and domain = '".$domain."'";
  $query = $this->db->query($sql);
  if($query->num_rows){
   return $query->row;
  } else {
   return "";
  }
 }
    public function cacheQuery($str, $qry, $json, $concepts = "", $domain = "BL", $schema = "")
    {
        $sql = "INSERT INTO sparql_cache VALUES('".$this->db->escape($str)."', '".$this->db->escape($qry)."', '".
                $this->db->escape($concepts)."','".$domain."','".$this->db->escape($schema)."',"
                . "'".$this->db->escape($json)."')";
        $query = $this->db->query($sql);
    }
}
```

Listing 20: "classes/query_generator.php"

```
<?php

class Query_generator
{
    private $db;
    public $domains = array();
    public $qualified_domains = array();
    public $user_query = "";
    private $cpi_dispatcher;
    private $cat; //cat obj
    private $cats; //store array of cats values
    private $sparql_queries = array();// to store generated SPARQL Queries
    public $statements = array();
    public $filters = array();
    private $domain = "";
    public $concepts;
    public $short_names = array();
    public $str = "";
    public $schema = array();
    public $results = array();

    public function __construct($db, $str)
    {
        $this->db = $db;
        $this->str = $str;
        $this->cpi_dispatcher = new Cpi_dispatcher($db);
        $this->cat = new Cat($db);
    }

    public function generateQueries()
    {
        $phrases = Syntax_analyser::getSyntax($this->user_query);
  foreach($this->qualified_domains as $domain){
            if(is_array($domain)) $domain = $domain[0];
            foreach($phrases as $phrase){

                $this->parsePhrases($phrase, $domain);
            }
        }
        $this->getRanks();
    }

    /*
     * get Ranks Algorithm
     */
    private function getRanks(){
        $class = $this->cats['cats']['class'];
        $class_query = "class='".$this->db->escape($class)."'";
        $sql = "SELECT cd.domain_id
                FROM classes c
                LEFT JOIN class_domains cd ON c.id = cd.class_id
                WHERE c.group_id IN (SELECT group_id FROM classes WHERE class = '".$class."') AND c.`schema` <> 'sem'
                ORDER BY c.usage_counter DESC";
        $query = $this->db->query($sql);
        $results = $query->rows;
        //pr($this->sparql_queries,1);
        foreach($results as $result){
            $this->executeSPARQL($this->sparql_queries[$result['domain_id']]);
        }
    }


    public function parsePhrases($phrase, $domain){
        $this->domain = $domain;
        $str = "";
        if(is_array($phrase)){
            if(isset($phrase["type"]) && $phrase["type"]=="and"){
                $this->processPhrase($phrase[0], $domain);
                $this->processPhrase($phrase[1], $domain);
```

```
        }
        else if(isset($phrase['case']) && $phrase['case'] == 'of'){
            unset($phrase['case']);
            $str = implode(" ", $phrase);
            $this->processPhrase($str, $domain);
        }
        else{
            $str = implode(" ", $phrase);
            $this->processPhrase($str, $domain);
        }
    }
    else{
        $this->processPhrase($phrase, $domain);
    }
}
public function processPhrase($phrase, $domain){
    //pr($this->domains,1);
    $words = Tokenizer::stringTokenizer($phrase);
    $parsed_words = array();
    foreach($words as $word1){
        if(!empty($this->domains[$domain]["words"][$word1]))
            $parsed_words[] = $this->domains[$domain]["words"][$word1];
    }
   // pr($parsed_words,1);
    foreach($parsed_words[0] as $word1){
      if(isset($parsed_words[1])){
        foreach($parsed_words[1] as $word2){
            $this->cpi_dispatcher->domain = $domain;
            $this->cpi_dispatcher->findRelations($word1, $word2);
        }
      } else {
        $this->cpi_dispatcher->domain = $domain;
        $this->cpi_dispatcher->findRelations($word1, "", true);
      }
    }
    $this->cpi_dispatcher->getRelations();
    $this->buildStatements($this->cpi_dispatcher->cpi_relations);
}
public function buildStatements($relations){
    $no_of_relations = count($relations);
    if($no_of_relations == 1){
        $relation = $relations[0];
        if($relation["type"]=="PI"){
            $stmts = $this->buildPIStatement($relation);
        }
        else if($relation["type"]=="CP"){
            $stmts = $this->buildCPStatement($relation);
        }
        else if($relation['type'] == "C"){
            $stmts = $this->buildCStatement($relation);
        }else if($relation['type'] == "P"){
            $stmts = $this->buildPStatement($relation);
        }else if($relation['type'] == "I"){
            $stmts = $this->buildIStatement($relation);
        }
        $this->statements[] = '{'.implode('', $stmts).'}';
        $this->cats = $this->cat->calculateCAT($relations, $this->concepts);
        if($this->cats['isclass']){ //if CAT return a class
            $catstatements = $this->buildCatStatement($this->cats);
            $this->statements[] = $catstatements;
        }
        $this->buildSPARQL();
    }
    else if($no_of_relations > 1){
        $i = 0;
        pr($relations);
        foreach($relations as $relation){
            if($relation["type"]=="CI"){
                $stmts = $this->buildCIStatement($relation);
            }
            else if($relation["type"]=="CP"){
                $stmts = $this->buildCPStatement($relation);
            }
```

```
            else if($relation['type'] == "C"){
                $stmts = $this->buildCStatement($relation);
            }
            else if($relation['type'] == "P"){
                $stmts = $this->buildPStatement($relation);
            }else if($relation['type'] == "I"){
                $stmts = $this->buildIStatement($relation);
            }

            if($i>0){
                $this->statements[] = htmlentities(' UNION {'.implode('', $stmts).'}');
            } else {
                $this->statements[] = htmlentities('{'.implode('', $stmts).'}');
            }
            $i++;
        }
        $this->cats = $this->cat->calculateCAT($relations, $this->concepts);
        if($this->cats['isclass']){ //if CAT return a class
            $catstatements = $this->buildCatStatement($this->cats);
            $this->statements[] = $catstatements;
        }
        $this->buildSPARQL();
    }
    else{
        // do nothing OR may be display some message
    }
}
public function buildCatStatement($cats){
    $schema_o = new Schema();
    $statements = "";
    $class = $cats['cats']['class'];
    $tokens = explode('/', $class);
    $var1 = $tokens[sizeof($tokens)-1];

    $var1 = $this->filterVars($var1);
    $statements[] = "{?".$var1." a <".$class.">}"; //create first variable for domain class e.g. ?book

    $schema = $schema_o->getSchema($var1, $this->domain);
    if($schema <> null){
        $this->schema[] = $var1.",".$schema[1];
        $statements = "{?".$var1." <".$schema[0]."> ?".$schema[1]."}";
    }
    return $statements;
}
public function buildPIStatement($relation){
    $search = array("{","}");
    $schema_o = new Schema();
    $statements = array();
    foreach($relation["relation"] as $r){
        $link = $r["links"];
        $link = str_replace($search, "", $link);
        $links = explode(",",$link);
        if(count($links) == 3){
            if($links[2]== "?"){
                // generate a variable
            }
            else{ // we are assuming it is a URI,
                $tokens = explode('/', $links[0]);
                $var1 = $tokens[sizeof($tokens)-1];

                $tokens_1 = explode('/', $links[2]);
                $var2 = $tokens_1[sizeof($tokens_1)-1];
                $var1 = $this->filterVars($var1);
                $var2 = $this->filterVars($var2);
                $statements[] = "{?".$var1." a <".$links[0].">}"; //create first variable for domain class e.g.
                    ?book
                $statements[] = "{?".$var2." a <".$links[2].">}"; //create second variable for range class e.g.
                    ?person

                $this->statements[] = "{?".$var1." <".$relation["property"]."> ?".$var2."}";
                $schema = $schema_o->getSchema($var1, $this->domain);
                if($schema <> null){
                    $this->schema[] = $var1.",".$schema[1];
```

```
                    $statements[] = "{?".$var1." <".$schema[0]."> ?".$schema[1]."}";
                }
                $schema1 = $schema_o->getSchema($var2, $this->domain);
                if($schema1 <> null){
                    $this->schema[] = $var2.",".$schema1[1];
                    $statements[] = "{?".$var2." <".$schema1[0]."> ?".$schema1[1]."}";
                }
            }
        }
    }
    // Now create statements for the instance
    $tokens_i = explode('/', $relation["instance"]["class"]);
    $var_i = $tokens_i[sizeof($tokens_i)-1];
    $tokens_p = explode('/', $relation["instance"]["property"]);
    $var_p = $tokens_p[sizeof($tokens_p)-1];
    $statements[] = "{?".$var_i." a <".$relation["instance"]["class"].">}";
    $statements[] = "{?".$var_i." <".$relation["instance"]["property"]."> ?".$var_p."}";
    $schema = $schema_o->getSchema($var_i, $this->domain);
    if($schema <> null){
        $this->schema[] = $var_i.",".$schema[1];
        $statements[] = "{?".$var_i." <".$schema[0]."> ?".$schema[1]."}";
    }

    $this->filters[] = "REGEX(?".$var_p.", '".$relation["instance"]["word"]."','i')";
    return $statements;
}
public function buildCIStatement($relation){
    $search = array("{","}");
    $schema_o = new Schema();
    $statements = array();
    foreach($relation["relation"] as $r){
        $link = $r["links"];
        $link = str_replace($search, "", $link);
        $links = explode(",",$link);
        if(count($links) == 3){
            if($links[2]== "?"){
                // generate a variable
            }
            else{ // we are assuming it is a URI
                $tokens = explode('/', $links[0]);
                $var1 = $tokens[sizeof($tokens)-1];

                $tokens_1 = explode('/', $links[2]);
                $var2 = $tokens_1[sizeof($tokens_1)-1];
                $var1 = $this->filterVars($var1);
                $var2 = $this->filterVars($var2);
                $statements[] = "{?".$var1." a <".$links[0].">}"; //create first variable for domain class e.g.
                    ?book
                $statements[] = "{?".$var2." a <".$links[2].">}"; //create second variable for range class e.g.
                    ?person

                $statements[] = "{?".$var1." <".$r["rel_property"]."> ?".$var2."}";
                $schema = $schema_o->getSchema($var1, $this->domain);
                if($schema <> null){
                    $this->schema[] = $var1.",".$schema[1];
                    $statements[] = "{?".$var1." <".$schema[0]."> ?".$schema[1]."}";
                }
                $schema1 = $schema_o->getSchema($var2, $this->domain);
                if($schema1 <> null){
                    $this->schema[] = $var2.",".$schema1[1];
                    $statements[] = "{?".$var2." <".$schema1[0]."> ?".$schema1[1]."}";
                }
            }
        }
    }
    // Now create statements for the instance
    $tokens_i = explode('/', $relation["instance"]["class"]);
    $var_i = $tokens_i[sizeof($tokens_i)-1];
    $tokens_p = explode('/', $relation["instance"]["property"]);
    $var_p = $tokens_p[sizeof($tokens_p)-1];
    $statements[] = "{?".$var_i." a <".$relation["instance"]["class"].">}";
    $statements[] = "{?".$var_i." <".$relation["instance"]["property"]."> ?".$var_p."}";
    $schema = $schema_o->getSchema($var_i, $this->domain);
```

```
    if($schema <> null){
        $this->schema[] = $var_i.",".$schema[1];
        $statements[] = "{?".$var_i." <".$schema[0]."> ?".$schema[1]."}";
    }
    $this->filters[] = "REGEX(?".$var_p.", '".$relation["instance"]["word"]."','i')";
    $statements = array_unique($statements);
    return $statements;
}
public function buildCPStatement($relation){
    $search = array("{","}");
    $schema_o = new Schema();
    $statements = array();
    foreach($relation["relation"] as $r){
        $link = $r["links"];
        $link = str_replace($search, "", $link);
        $links = explode(",",$link);
        if(count($links) == 3){
            if($links[2]== "?"){
                // generate a variable
            }
            else{ // we are assuming it is a URI,
                $tokens = explode('/', $links[0]);
                $var1 = $tokens[sizeof($tokens)-1];

                $tokens_1 = explode('/', $links[2]);
                $var2 = $tokens_1[sizeof($tokens_1)-1];
                $var1 = $this->filterVars($var1);
                $var2 = $this->filterVars($var2);
                $statements[] = "{?".$var1." a <".$links[0].">}"; //create first variable for domain class e.g.
                    ?book
                $statements[] = "{?".$var2." a <".$links[2].">}"; //create second variable for range class e.g.
                    ?person

                $statements[] = "{?".$var1." <".$r["rel_property"]."> ?".$var2."}";
                $schema = $schema_o->getSchema($var1, $this->domain);
                if($schema <> null){
                    $this->schema[] = $var1.",".$schema[1];
                    $statements[] = "{?".$var1." <".$schema[0]."> ?".$var1."_".$schema[1]."}";
                }
                $schema1 = $schema_o->getSchema($var2, $this->domain);
                if($schema1 <> null){
                    $this->schema[] = $var2.",".$schema1[1];
                    $statements[] = "{?".$var2." <".$schema1[0]."> ?".$var2."_".$schema1[1]."}";
                }
            }
        }
    }
    $statements = array_unique($statements);
    pr($statements);
    return $statements;
}

/*
 * prepare statement for only class
 */
public function buildCStatement($relation){
    $statements = array();
    $search = array("{","}");
    $schema_o = new Schema();
    $link = $relation['class']['class'];
    $tokens = explode('/', $link);
    $var = $tokens[sizeof($tokens)-1];
    $var = $this->filterVars($var);
    $statements[] = "{?".$var." a <".$link.">}"; //create variable for domain class e.g. ?book
    $schema = $schema_o->getSchema($var, $this->domain);
    if($schema <> null){
        $this->schema[] = $var.",".$schema[1];
        $statements[] = "{?".$var." <".$schema[0]."> ?".$schema[1]."}";
    }
    return $statements;
}

/*
```

```
 * Prepare statement for only property
 */
public function buildPStatement($relation){
    $search = array("{","}");
    $statements = array();
    $schema_o = new Schema();
    foreach($relation["relation"] as $r){
        $link = $r["links"];
        $link = str_replace($search, "", $link);
        $links = explode(",",$link);
        if(count($links) == 3){
            if($links[2]== "?"){
                // generate a variable
            }
            else{ // we are assuming it is a URI
                $tokens = explode('/', $links[0]);
                $var1 = $tokens[sizeof($tokens)-1];

                $tokens_1 = explode('/', $links[2]);
                $var2 = $tokens_1[sizeof($tokens_1)-1];
                $var1 = $this->filterVars($var1);
                $var2 = $this->filterVars($var2);
                $statements[] = "{?".$var1." a <".$links[0].">}"; //create first variable for domain class e.g.
                    ?book
                $statements[] = "{?".$var2." a <".$links[2].">}"; //create second variable for range class e.g.
                    ?person

                $statements[] = "{?".$var1." <".$r["rel_property"]."> ?".$var2."}";
                $schema = $schema_o->getSchema($var1, $this->domain);
                if($schema <> null){
                    $this->schema[] = $var1.",".$schema[1];
                    $statements[] = "{?".$var1." <".$schema[0]."> ?".$schema[1]."}";
                }
                $schema1 = $schema_o->getSchema($var2, $this->domain);
                if($schema1 <> null){
                    $this->schema[] = $var2.",".$schema1[1];
                    $statements[] = "{?".$var2." <".$schema1[0]."> ?".$schema1[1]."}";
                }
                // $this->statements[] = "{?".$var1." <".$relation["property"]."> <".$links[2].">}";
            }
        }
    }
    if(!$relation['relation']){
        $link = $relation['class']['property'];
        $links[0] = $relation['class']['domain'];
        $links[1] = $relation['class']['range'];
        if($links[0]) {
            $tokens = explode('/', $links[0]);
            $var1 = $tokens[sizeof($tokens)-1];
        } else {
            $var1 = "var1";
        }
        if($links[1]) {
            $tokens = explode('/', $links[1]);
            $var2 = $tokens[sizeof($tokens)-1];
        } else {
            $var2 = "var2";
        }
        $var1 = $this->filterVars($var1);
        $var2 = $this->filterVars($var2);
        if($links[0])
        $statements[] = "{?".$var1." a <".$links[0].">}"; //create first variable for domain class e.g. ?book
        if($links[1])
        $statements[] = "{?".$var2." a <".$links[1].">}"; //create second variable for range class e.g. ?person

        $statements[] = "{?".$var1." <".$link."> ?".$var2."}";
        $schema = $schema_o->getSchema($var1, $this->domain);
        if($schema <> null){
            $this->schema[] = $var.",".$schema[1];
            $statements[] = "{?".$var." <".$schema[0]."> ?".$schema[1]."}";
        }
    }
```

```
        return array_unique($statements);
    }

    /*
     * prepare statement for only instance
     */
    public function buildIStatement($relation){
        $schema_o = new Schema();
        $tokens_i = explode('/', $relation["instance"]["class"]);
        $var_i = $tokens_i[sizeof($tokens_i)-1];
        $tokens_p = explode('/', $relation["instance"]["property"]);
        $var_p = $tokens_p[sizeof($tokens_p)-1];
        $statements[] = "{?".$var_i." a <".$relation["instance"]["class"].">}";
        $statements[] = "{?".$var_i." <".$relation["instance"]["property"]."> ?".$var_p."}";
        $schema = $schema_o->getSchema($var_i, $this->domain);
        if($schema <> null){
            $this->schema[] = $var_i.",".$schema[1];
            $statements[] = "{?".$var_i." <".$schema[0]."> ?".$schema[1]."}";
        }
        return $statements;
    }
    /**
     *buildSPARQL function
     */
    public function buildSPARQL(){
        echo "Echo Build Statement Called";
        $this->statements = array_unique($this->statements);
        $conditions = new Conditions();
        $this->statements = $conditions->handleConjunctions($this->statements);
        $this->statements =$conditions->handleNegation($this->statements);
        $this->statements = $conditions->handleQuantification($this->statements);
        $sparql = " SELECT * ".
                  " WHERE {";
        $sparql .=  implode("", $this->statements);
        if(count($this->filters) > 0) {
            $sparql .= "FILTER ( ";
            $sparql .= implode(" || ", $this->filters);
            $sparql .=  " )";
        }
        $sparql .= "}";
        $sparql .= "LIMIT 10";
        $this->sparql_queries[$this->domain] = $sparql;
        //echo htmlentities($sparql);


    }

    public function executeSPARQL($sparql) {
        $this->schema = array_unique($this->schema);
        $schemas = implode(",",$this->schema);
        $dm = new  Domains($this->db);
        $endpoint = $dm->getEndPoint($this->domain);
        $api = new  API($endpoint);
        $results = $api->getResults($sparql);
        $qf = new Query_formatter($this->db);
        $qf->cacheQuery($this->str, $sparql, json_encode($results),"",$this->domain,$schemas);
        $this->results[] = $results;
        //print_r($results);
    }
    public function filterVars($var) {
        $var = str_replace(array('#'),array(''), $var);
        return $var;
    }
}
```

Listing 21: "classes/schema.php"

```
<?php
/**
 * Created by PhpStorm.
```

```
 * User: Arooj
 * Date: 8/8/2015
 * Time: 10:44 PM
 */
class Schema
{
    public  $schema = array();

    public function getSchema($concept, $domain)
    {
        if($concept == "Book" && $domain==1){
            $schema = array("http://purl.org/dc/terms/title", "title");
        }
        else if($concept == "Person" && $domain==1){
            $schema = array("http://xmlns.com/foaf/0.1/name", "name");
        }
        else if($concept == "Agent" && $domain==1){
            $schema = array("http://xmlns.com/foaf/0.1/name", "name");
        }
        else{
            $schema = null;
        }
        return $schema;
    }
}
```

Listing 22: "classes/sem_properties.php"

```
<?php
class Sem_Properties{
 private $db;
 private $domain = "http://semantira.com/";
 private $schema = "sem";
 public function __construct($db){
    $this->db= $db;
  $this->addSemproperties();
  }

 public function addSemProperties(){
  $sql = "SELECT * FROM properties GROUP BY property_group";
  $results = $this->db->query($sql);
  $properties =  $results->rows;
  foreach($properties as $property) {
   $prop_short = explode(":", $property['property_short']);

   echo "<br/>".$insert = "INSERT INTO properties  SET property='".$this->domain.$prop_short[1]."', property_short =
        '".$this->schema.":".$prop_short['1']."', domain ='".$property['domain']."',
       property_range='".$property['property_range']."', property_type ='".$property['property_type']."',
       property_group ='".$property['property_group']."'";
   $this->db->query($insert);
  }
  echo $totalUpdated." ".$this->schema ." properties added into system.";

 }
}
```

Listing 23: "classes/semantic_analyser.php"

```
<?php

class Semantic_analyser{
    private $db;
    public  $classes=  array();
    public  $properties=  array();
    public  $instances=  array();
    public  $concepts =  array();
    private $words;
 private $domains, $qg;
```

```
public $qualified_domains = array();
public function __construct($db, $str){
    $this->db= $db;
    $this->domains = new Domains($db);
    $this->qg = new Query_generator($db, $str);
    $this->processWords($str); // get all concepts
}
public  function parseSyntax($phrases=array()){

}
public function processWords($phrase){
    $this->words= array();
    $this->words = Tokenizer::stringTokenizer($phrase);
    $this->getClasses(); // classes
    $this->getProperties();
    $this->getInstances();
    $this->qualified_domains = $this->domains->getMappedDomains($this->concepts);
    $this->qg->qualified_domains = $this->qualified_domains;
    $this->domains->getRankedDomains();
    $this->qg->domains = $this->domains->domains;
    $this->qg->user_query = $phrase; // assign phrase to QG class to generate queries
    $this->qg->concepts = $this->concepts;
    $this->qg->generateQueries();
    //$this->getCPITags();
}
public function getClasses(){
    foreach($this->words as $word){
        if(trim($word) <>""){
            $sql = "SELECT `class`, `title`, `schema`
   FROM classes
   WHERE title='".$this->db->escape($word)."'
    OR class IN (SELECT class FROM class_map WHERE term='".$this->db->escape($word)."')
                        GROUP BY group_id";
            $query = $this->db->query($sql);
            if($query->num_rows > 0){
                foreach($query->rows as $row){
  $this->concepts["classes"][] = array("type"=>"C", "word"=>$word, "class"=>$row["class"]);
                }
            }
        }
    }
   // $this->classes = array_map("unserialize", array_unique(array_map("serialize", $this->classes)));
   // echo "Classes Found <pre>"; print_r($this->classes); echo "</pre>";
}
public function getProperties(){
    foreach($this->words as $word){
        if(trim($word) <>""){
            $sql = "SELECT p.property, IF( p.class =  '', p.domain, p.class ) AS final_class, p.class, p.domain,
         p.property_range, p.property_short, p.property_type,  c.title, c.`schema`
   FROM properties p LEFT JOIN classes c ON ( p.class = c.class OR p.domain = c.class )
   WHERE  p.property LIKE '%/".$this->db->escape($word)."'
       OR p.property IN (SELECT property FROM property_map WHERE term='".$this->db->escape($word)."')
                   GROUP BY p.property";
            $query = $this->db->query($sql);
            if($query->num_rows > 0){
                foreach($query->rows as $row){
  $this->concepts["properties"][] = array("type"=>"P", "word"=>$word, "property"=>$row["property"]);
                }
            }
        }
    }
}
public function getInstances(){
    foreach($this->words as $word){
        $sql = "SELECT bw.term, bw.property, bw.property_uri, bw.rdf_type, bw.explicit, p.property_range,
             p.property_type
 FROM bag_of_words bw
      LEFT JOIN properties p ON bw.property_uri = p.property
    WHERE term = '".$this->db->escape($word)."'";
        $query = $this->db->query($sql);

        $instance_array_per_word = array();
        $object_property_related = array();
```

```
        if($query->num_rows > 0 && trim($word)<>""){
            foreach($query->rows as $row){
 $classes = explode(",",$row["rdf_type"]);
                $classes = array_unique($classes);
 $this->concepts["instances"][] = array("type"=>"I", "word"=>$word, "class"=>$classes,
                                                        "property"=>$row["property_uri"]);
            }
        }
    }
}
public function getCPITags(){
    $class_property = array();
    $property_instance = array();
    $class_instance = array();
    foreach($this->classes as $class){
        foreach($this->properties as $property){
            $result = $this->isPropertyRelatedToClass($property["property"], $class["class"]);
            if(count($result) > 0){
                $class_property[$class["class"]] = array("class"=>$class["class"],
                    "property"=>$property["property"],
                    "relation"=>$result);
            }

        }
    }
    echo 'Class-Property Relations: <pre>';
    print_r($class_property);
    echo '</pre>';
    foreach($this->properties as $property){
        foreach($this->instances as $instances){
            foreach($instances as $instance){
                if(isset($instance["property_uri"]) && $property["property"] == $instance["property_uri"]){
                    $property_instance[] = array("instance"=>$instance["word"],
                        "property"=>$property["property"],
                        "class"=>$instance["class"],
                        "domain"=>$property["domain"],
                        "range"=>$property["range"],
                        "relation"=>"direct");
                }
                else if( $property["actual_class"] == $instance["class_uri"]){
                    $property_instance[] = array("instance"=>$instance["word"],
                        "property"=>$property["property"],
                        "class"=>$instance["class"],
                        "domain"=>$property["domain"],
                        "range"=>$property["range"],
                        "relation"=>"indirect_class");

                }
                else if( $property["domain"] == $instance["class_uri"]){
                    $property_instance[] = array("instance"=>$instance["word"],
                        "property"=>$property["property"],
                        "class"=>$instance["class"],
                        "domain"=>$property["domain"],
                        "range"=>$property["range"],
                        "relation"=>"indirect_domain");

                }
                else if( $property["range"] == $instance["class_uri"]){
                    $property_instance[] = array("instance"=>$instance["word"],
                        "property"=>$property["property"],
                        "class"=>$instance["class"],
                        "domain"=>$property["domain"],
                        "range"=>$property["range"],
                        "relation"=>"indirect_range");

                }
            }
        }
    }
    echo 'Property- Instance Relations: <pre>';
    print_r($property_instance);
    echo '</pre>';
    foreach($this->classes as $class){
```

```
            foreach($this->instances as $instances){
                foreach($instances as $instance){
                    if($class["class"] == $instance["class"]){
                        $data = array("instance"=>$instance["word"],
                            "property"=>$instance["property"],
                            "class"=>$instance["class"]);
                        $class_instance[] = $data;
                        $class_property[$instance["class"]]["instance"] = $data;
                    }
                }
            }
        }
        echo 'Class- Instance Relations: <pre>';
        print_r($class_instance);
        echo '</pre>';

        echo 'Class-Property Relations: <pre>';
        print_r($class_property);
        echo '</pre>';

    }
    private function isPropertyRelatedToClass($property, $class){
        $sql = "SELECT property, class, domain, property_range FROM properties WHERE
             (property='".$this->db->escape($property)."' AND class='".$this->db->escape($class)."')
        OR (property='".$this->db->escape($property)."'
          AND (domain='".$this->db->escape($class)."'
        OR property_range='".$this->db->escape($class)."')
        )";
        $query = $this->db->query($sql);
        if($query->num_rows > 0){
            $prop_classes = array();
            foreach($query->rows as $row){
                if($row["class"] == ""){
                    $row["class"] = $row["domain"];
                }
                $prop_classes[] = array("domain"=>$row["class"], "property"=>$row["property"],
                     "range"=>$row["property_range"]);
            }
            return $prop_classes;
        }
        return NULL;
    }
}
```

Listing 24: "classes/stack.php"

```
<?php
class Stack{
 private $data = array();
 private $stack = array();
 private $joinStack = array();
 private $conditionStack = array();
 private $ignore = array('root', 'nmod:of');
 private $ignoreConditions = array('in', 'of', 'with');
 public function __construct($data){
  $this->data = $data;
 }
 /*
 *Following relation belongs to left arc
 *det,ref,acl
 */
 public function leftArc($set,$arr =true) { //left arc
  $return = $set[1]['feature'].' '.$set[0]['feature'];
  if($arr) {
   return array($set[0]['index'] => $set[0]['feature']);
  }
  else
  return $return;
 }
 /*
```

```
*Following relation belongs to left arc
*
*dobj
*/
public function rightArc($set,$arr =true) { //right arc
 $return = $set[1]['feature'].' '.$set[0]['feature'];
 if($arr){
  return  array($set[1]['index'] => $set[1]['feature']);
 }
 else
 return $return;
}

/*
*
*/
public function createStack($data= array()) {
 $stack = array();
 $len = sizeof($data);
 $lastStep = "";
 for($i=0; $i<$len;  $i++){
  $set = $data[$i];
  if(in_array($set['type'],array('cop'))){ //join
   $arr = array($set[0]['index'] =>$set[0]['feature'], $set[1]['index'] => $set[1]['feature']);
   ksort($arr);
   $stack[$set[0]['index']] = array(implode(",",$arr));
  } else if(in_array($set['type'], array('det'))){ //left Arch
   $stack[$set[0]['index']] = array(implode(",",$this->leftArc($set)));
  } else if(in_array($set['type'], array('ref'))){ //right Arch
   $stack[$set[1]['index']] = array(implode(",",$this->leftArc($set)));
  } else if($set['type'] == 'compound') {
   $arr = array();
   if(isset($stack[$set[0]['index']])) {
    $arr = array($set[0]['index']=>$set[0]['feature'], $set[1]['index']=>$set[1]['feature']);
    $arr = array_merge($arr, $stack[$set[0]['index']]);
   }else{
    $arr = array($set[0]['index']=>$set[0]['feature'], $set[1]['index']=>$set[1]['feature']);
   }
   ksort($arr);
   $stack[$set[0]['index']] = $arr;
  }
  else {
   if(!isset($stack[$i])){
    //$stack[$set[0]['index']] = $set;
   }
  }
 }
 $this->stack = $stack;
 return $this->stack;
}

public function applyConditions($data=array(), $output = array()) {
 $final = array();
 $stack = $output;
 $len = sizeof($data);
 for($i=0; $i<$len;  $i++){
  $set = $data[$i];
  if($set['type'] == 'nsubj'){ //if subject
   //$final[$set[0]['index']] = $output[$set[0]['index']];
   //$final[$set[1]['index']] = $output[$set[1]['index']];
  } else if($set['type'] == 'case'){ //if condition
   $arr1 = $arr2 = array();

   $arr1=isset($output[$set[1]['index']-1]) ? $output[$set[1]['index']-1] :
        array($data[$set[1]['index']-1][1]['feature']);
   if(isset($arr1[0]) && self::findKeyValue($arr1[0], $final) != '') {
    $arr1 = array();
   }else if(isset($arr1[0]) && in_array($arr1[0], $this->ignoreConditions)) {
    $arr1 = array();
   }

   $stack = self::removeElement($set[1]['index']-1,$stack);
```

```
   $arr2=isset($output[$set[0]['index']]) ? $output[$set[0]['index']] : array($set[0]['feature']);
   if(isset($arr2[0]) && self::findKeyValue($arr2[0], $final) != '') {
    $arr2 = array();
   } else if(isset($arr2[0]) && in_array($arr2[0], $this->ignoreConditions)) {
    $arr2 = array();
   }
   $stack = self::removeElement($set[0]['index'],$stack);

   $final[$set[0]['index']] = array_merge($arr1, array('case'=>$set[1]['feature']), $arr2);

  } else if(in_array($set['type'],array('dobj'))){ //if dobj
   if(isset($output[$set[1]['index']])) {
    $arr= $output[$set[1]['index']];
    unset($stack[$set[1]['index']]);
   } else {
    $arr= array($set[1]['index'] => $set[1]['feature']);
   }

   $final[$set[1]['index']] = array_merge(array($set[0]['index']=>$set[0]['feature']), $arr);

  } else if(in_array($set['type'],array('acl'))){ //if acl
   if(isset($final[$set[0]['index']])) {
    $arr = $final[$set[0]['index']];
    unset($final[$set[0]['index']]);
   } else{
    $arr=isset($output[$set[0]['index']]) ? $output[$set[0]['index']] : array($set[0]['index'] => $set[0]['feature']);
   }

   $final[$set[1]['index']] = array_merge($arr, array($set[1]['index'] => $set[1]['feature']));
   ksort($final[$set[1]['index']]);
  }
 }
 return $this->joinStack = array('stack'=>$stack, 'conditions'=>$final);
}

public function applyJoins($data =array(), $output = array()) {
 $final = array();
 $stack = $output['stack'];
 $conditions = $output['conditions'];
 $len = sizeof($data);
 for($i=0; $i<$len;  $i++){
  $set = $data[$i];
  if($set['type'] == 'conj:and'){ //if and
   $arr1 = $arr2 = array();

   $cindex1 = self::findKeyValue($set[0]['feature'], $conditions);
   $cindex2 = self::findKeyValue($set[1]['feature'], $conditions);
   if($cindex1 != '') {
    $arr1 = implode(' ', $conditions[$cindex1]);
    unset($conditions[$cindex1]);
   }
   if($cindex2 != '') {
    $arr2 = implode(' ', $conditions[$cindex2]);
    unset($conditions[$cindex2]);
   }
   $index = max($cindex1, $cindex2);
   $final[$index] = array('type'=>'and', $arr1, $arr2);
  }
 }
 return $this->joinStack = array( 'joins' => $final, 'conditions'=>$conditions, 'stack'=>$stack);
}

public function createOuput($data = array()) {
 $stack = $data['stack'];
 $conditions = $data['conditions'];
 $joins = $data['joins'];
 $output = array();
 foreach($stack as $k=>$v) {
  $output[$k] = implode(' ', $v);
 }
 foreach($conditions as $k=>$v) {
  $output[$k] = implode(' ', $v);
 }
```

```
  foreach($joins as $k=>$v) {
   $output[$k] =  $v;
  }
  ksort($output);
  return $output;
 }
 public function removeElement($key, $data) {
  if(isset($data[$key])) {
   unset($data[$key]);
  }
  return $data;
 }
 public function findKeyValue($key, $data = array(), $debug = false) {
  if($debug) {
   echo "<br>======================";
   echo "key=".$key."<br>";
   print_r($data);
  }
  $return = '';
  foreach($data as $k=>$item) {
   if(is_array($item)){
    $find = array_search($key, $item);
    if($debug) { echo $find; print_r($item);}
    if($find !== false){$return = $k;}
   }
  }

  if($debug){
   echo "Return = $return, <br>======================<br>";
  }
  return $return;
 }

}
```

Listing 25: "classes/syntax_analyser.php"

```
<?php
class Syntax_analyser{
    public static $output = array();
 public static function getSyntax($str){
        if(self::$output) {
            return self::$output;
        }
  $parser = new \StanfordNLP\Parser(
        $_SERVER["DOCUMENT_ROOT"].'/src/stanford-parser.jar',
        $_SERVER["DOCUMENT_ROOT"].'/src/stanford-parser-3.5.2-models.jar'
     );
  $result = $parser->parseSentence($str);

        $data = $result['typedDependencies'];
  $depObj = new Dep_Parser($data);
  $output = $depObj->getOuput();
        self::$output = $output;
        return $output;
 }
}
```

Listing 26: "classes/tokenizer.php"

```
<?php
class Tokenizer {
 public function __construct($db){

 }
 public static function stringTokenizer($str) {
  if(is_array($str)) $str = implode(" ", $str);
  $find = array("-", 'first name', 'last name', 'family name','list');
```

```
  $replace = array("_", 'first_name', 'last_name', 'family_name','');
  $str = str_replace($find, $replace, $str);
  preg_match_all('/\w+|"[\w\s]*"/', $str, $matches);
  $words = array();
  foreach($matches[0] as $word){
    $words[] = ltrim(rtrim(str_replace(array('"', '_'),array("", "-"),$word)));
  }
  return $words;
 }
    public static function questionIdentifier($str){
  if(is_array($str)) $str = implode(" ", $str);
        if (strpos($str,'what') !== false ||
            strpos($str,'how') !== false ||
            strpos($str,'where') !== false) {
            return 'true';
        }
        return false;
    }
}
?>
```

Appendix III - List of Test Queries

No.	Query
01	What is the first name of the author and surname of the editor of Da Vinci?
02	List all authors born in 1945
03	Which titles by detective writer Ian Rankin appear in BNB?
04	Find books published in York
05	Search for a book with ISBN 9780729408745
06	Which states border Texas?
07	Horror movies
08	Find famous english songs
09	Population of cities california
10	Which city is the largest city in Pakistan?
11	Highest point of state bordering Mississippi
12	Who is the author of Pride and Prejudice?
13	List books published after 2011
14	Find authors died between 1977 and 1983
15	Simon's books published in 1972
16	Serial title with ISSN 0955-6664
17	Find authors whose name is hardy but surname is not tim
18	Find authors whose name is hardy or tim
19	Find person whose name is john or jack
20	Find books whose price are less than 50
21	List authors of physics
22	Book authors
23	Persons name
24	Stanislaw Piotowicz
25	Books
26	Authors
27	What is the currency of Pakistan
28	Which cities are located in Europe
29	List all shows by disney
30	Who is the writer of The Secret Life of Arabia?
31	What is the first name and surname of the author of Ride a Rhino?
32	Who is the prime minister of USA?
33	Neilsen's articles
34	List 50 persons born in 1992
35	List persons whose age is above 50

36	Find books on crystallography
37	List books for Mathematics
38	find cities and their population
39	What is the population of pakistan
40	Lord of the Rings
41	Who is the director of Da vinci?
42	The Da Vinci code
43	Cities in pakistan
44	What is the population of each city in pakistan?
45	List authors who published less than two books
46	Simon's books
47	Books published between 1972 and 1985
48	List 10 highest mountains in the world
49	List films directed by Shane Meadows
50	Which city of UK has the highest population?

Test User Queries (Source: Author, 2015)

Appendix IV - List of Software Versions Used

Software/Apps	**Version**
PHP-Stanford-NLP	3.5.2
Apache Jena	2.11.2
Apache Jena RDF API	2.11.2
Java API for WordNet Searching (JAWS)	3.0
Java Inflector library	2.1
Apache Jena Ontology API	2.0
OWL	1.1
SPARQL	1.0 and 1.1
MySQL	5.6
WAMP	5.5
PHP	5.5

List of Software Versions (Source: Author, 2015)

Appendix V - Screenshots of Back-end Server Processing

Ontology Processing

Ontology Parsing

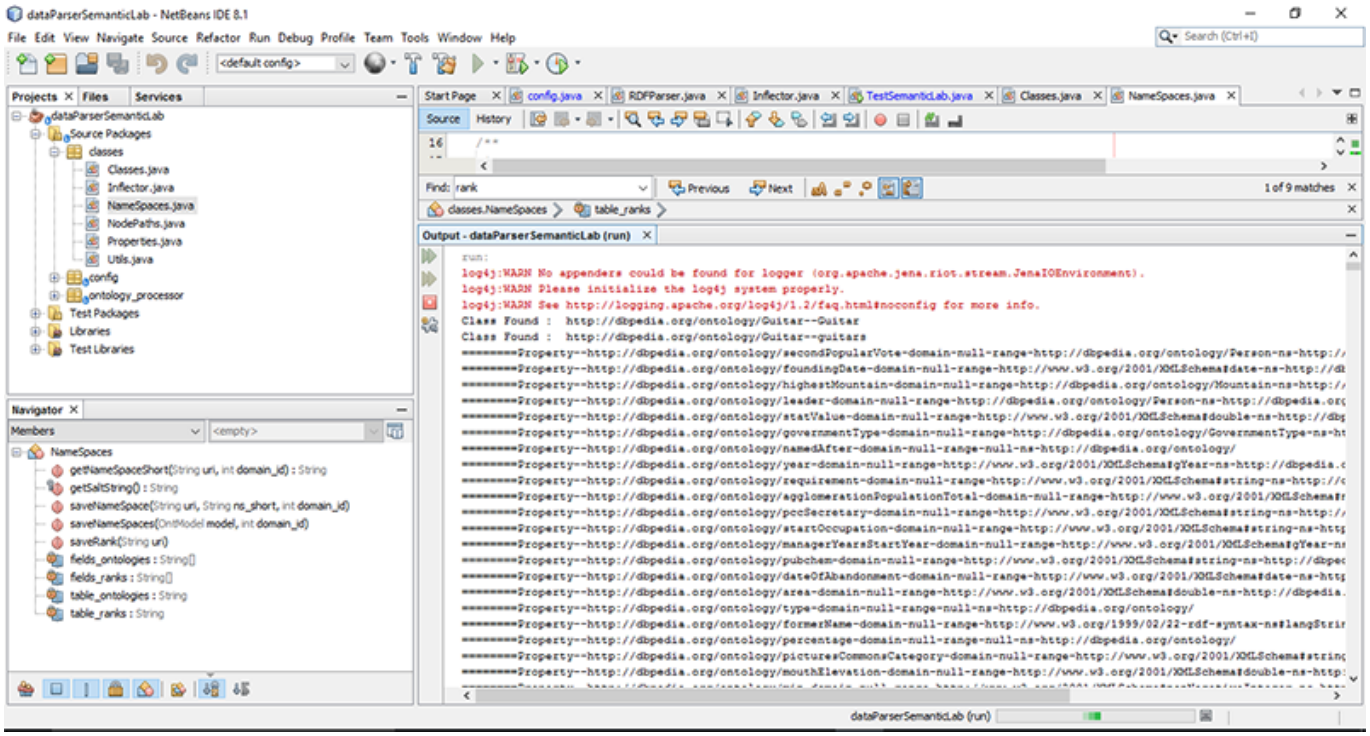


Figure 1: Running Code for Ontology Processing

Pre-computing ontology concepts from the given namespaces

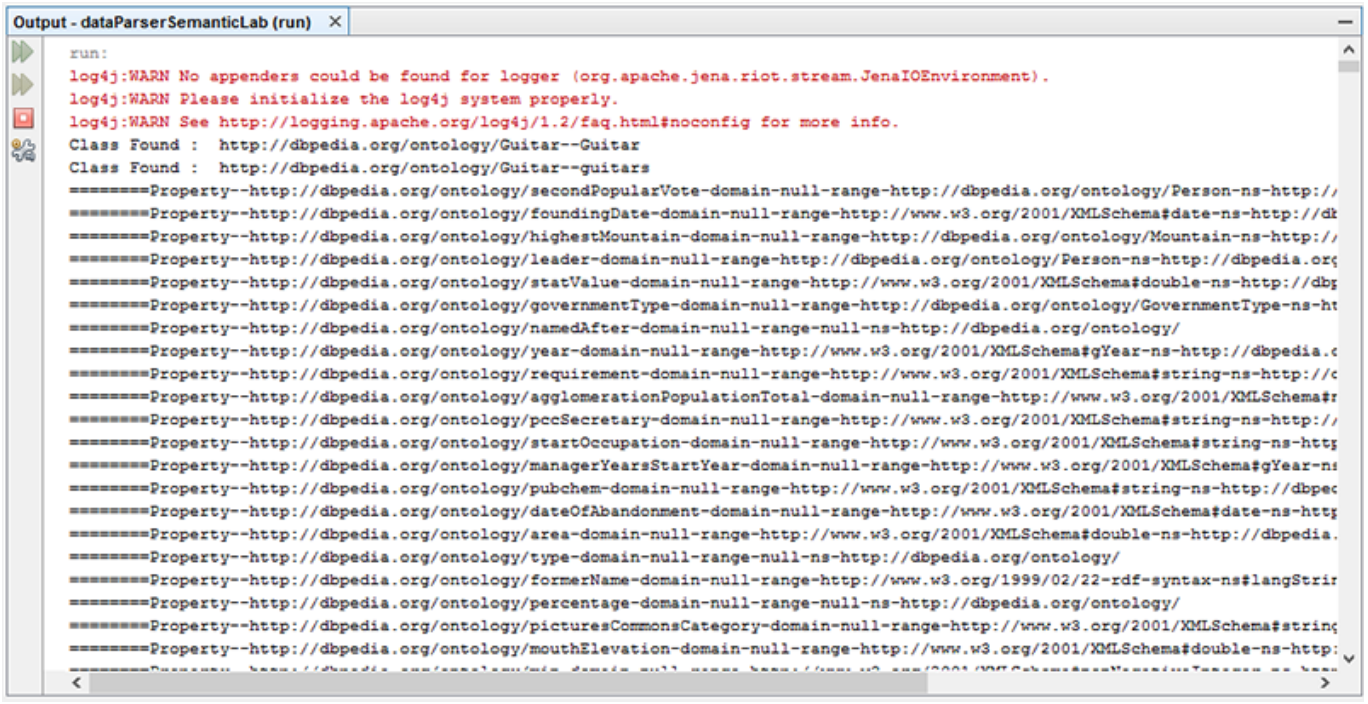


Figure 2: Output for Pre-computing Ontology Concepts

Saving concepts (i.e. classes and properties) in the database

id	class	title	schema	group_id	usage_counter
1	http://dbpedia.org/ontology/Guitar	Guitar	dbo	2	0
2	http://dbpedia.org/ontology/ProtectedArea	ProtectedArea	dbo	3	0
3	http://dbpedia.org/ontology/Galaxy	Galaxy	dbo	4	0
4	http://dbpedia.org/ontology/SoftballLeague	SoftballLeague	dbo	5	0
5	http://dbpedia.org/ontology/GovernmentAgency	GovernmentAgency	dbo	6	0
6	http://schema.org/GovernmentOrganization	GovernmentOrganization	dbo	6	0
7	http://dbpedia.org/ontology/Brewery	Brewery	dbo	7	0
8	http://dbpedia.org/ontology/RallyDriver	RallyDriver	dbo	8	0

Figure 3: Classes Table

Keyword Mapping (to their plural forms and synonyms)

```
SELECT * FROM `class_map`
```

Show all | Number of rows: 25 Filter rows: Search this table

Sort by key: None

+ Options

	id	class	term
Edit Copy Delete	10	http://dbpedia.org/ontology/Brewery	breweries
Edit Copy Delete	9	http://dbpedia.org/ontology/Brewery	brewery
Edit Copy Delete	8	http://dbpedia.org/ontology/Galaxy	galaxies
Edit Copy Delete	7	http://dbpedia.org/ontology/Galaxy	galaxy
Edit Copy Delete	1	http://dbpedia.org/ontology/Guitar	guitar
Edit Copy Delete	2	http://dbpedia.org/ontology/Guitar	guitars

Figure 4: Class Map Table

property	property_short	term
http://dbpedia.org/ontology/creator	APBDNK:creator	creator
http://dbpedia.org/ontology/creator	APBDNK:creator	creators
http://purl.org/dc/terms/creator	dct:creator	author
http://purl.org/dc/terms/creator	dct:creator	authors
http://purl.org/dc/terms/creator	dct:creator	writer
http://purl.org/dc/terms/creator	dct:creator	writers

Figure 5: Property Map Table

Data Parsing to create Bag-of-Keywords

term	property	property_uri	rdf_type
519.82^^http://dewey.info/schema-terms/Notation	skos:notation	http://www.w3.org/2004/02/skos/core#notation	http://www.w3.org/2004/02/skos/core#Concept, http...
Stanislaw	foaf:givenName	http://xmlns.com/foaf/0.1/givenName	http://purl.org/dc/terms/Agent, http://xmlns.com/...
Piotowicz	foaf:familyName	http://xmlns.com/foaf/0.1/familyName	http://purl.org/dc/terms/Agent, http://xmlns.com/...
Stanislaw Piotowicz	foaf:name	http://xmlns.com/foaf/0.1/name	http://purl.org/dc/terms/Agent, http://xmlns.com/...
Piotowicz, Stanislaw	rdfs:label	http://www.w3.org/2000/01/rdf-schema#label	http://purl.org/dc/terms/Agent, http://xmlns.com/...
Edinburgh : Oliver & Boyd, 1984	rdfs:label	http://www.w3.org/2000/01/rdf-schema#label	http://www.bl.uk/schemas/bibliographic/blterms#Pu...
Santa Monica, Calif. ; Great Britain : RAND, Scien...	rdfs:label	http://www.w3.org/2000/01/rdf-schema#label	http://www.bl.uk/schemas/bibliographic/blterms#Pu...
Nerve block	rdfs:label	http://www.w3.org/2000/01/rdf-schema#label	http://www.w3.org/2004/02/skos/core#Concept, http...
English objective tests	dct:title	http://purl.org/dc/terms/title	http://purl.org/ontology/bibo/Book, http://purl.o...
0333147642	bibo:isbn10	http://purl.org/ontology/bibo/isbn10	http://purl.org/ontology/bibo/Book, http://purl.o...
104p.@en	isbd:P1053	http://iflastandards.info/ns/isbd/elements/P1053	http://purl.org/ontology/bibo/Book, http://purl.o...
GB7301854	blt:bnb	http://www.bl.uk/schemas/bibliographic/blterms#bnb	http://purl.org/ontology/bibo/Book, http://purl.o...
English objective tests / David Self	rdfs:label	http://www.w3.org/2000/01/rdf-schema#label	http://purl.org/ontology/bibo/Book, http://purl.o...
China--Economic conditions	rdfs:label	http://www.w3.org/2000/01/rdf-schema#label	http://www.w3.org/2004/02/skos/core#Concept, http...

Figure 6: Concept Mapping

Back-end Query Formation

Example 1

Query 1:

"simon's books"

Syntax Analysis:

```
Array
(
    [2] => Array
        (
            [1] => simon
            [2] => books
        )

)
```

Figure 7: Query 1 - Syntax Analysis

Concept Mapping:

```
Classes Found
Array
(
    [0] => Array
        (
            [type] => C
            [word] => books
            [class] => http://purl.org/ontology/bibo/Book
        )
```

```
)
Array
(
    [0] => Array
        (
            [type] => CI
            [class] => http://purl.org/ontology/bibo/Book
            [instance] => Array
                (
                    [type] => I
                    [word] => simon
                    [class] => http://xmlns.com/foaf/0.1/Agent
                    [property] => http://xmlns.com/foaf/0.1/familyName
                )

            [relation] => Array
                (
                    [0] => Array
                        (
                            [rel_property] =>
                            [class] => http://purl.org/ontology/bibo/Book
                            [domain] => http://purl.org/ontology/bibo/Book
                            [range] => http://xmlns.com/foaf/0.1/Agent
                            [links] => http://purl.org/ontology/bibo/Book,blt:publication,http://www.bl.uk/schemas/bibliographic/blterms#PublicationEvent}
                                       {http://www.bl.uk/schemas/bibliographic/blterms#PublicationEvent,event:agent,http://xmlns.com/foaf/0.1/Agent},
                                       {http://purl.org/ontology/bibo/Book,blt:publication,http://www.bl.uk/schemas/bibliographic/blterms#PublicationEvent}
                                       {http://www.bl.uk/schemas/bibliographic/blterms#PublicationEvent,event:agent,http://purl.org/dc/terms/Agent
                            [rel_property_name] =>
                        )

                )

        )

    [1] => Array
        (
            [type] => CI
            [class] => http://purl.org/ontology/bibo/Book
            [instance] => Array
                (
                    [type] => I
                    [word] => simon
                    [class] => http://xmlns.com/foaf/0.1/Person
                    [property] => http://xmlns.com/foaf/0.1/familyName
                )

            [relation] => Array
                (
                    [0] => Array
                        (
                            [rel_property] =>
                            [class] => http://purl.org/ontology/bibo/Book
                            [domain] => http://purl.org/ontology/bibo/Book
                            [range] => http://xmlns.com/foaf/0.1/Person
                            [links] => http://purl.org/ontology/bibo/Book,dct:creator,http://xmlns.com/foaf/0.1/Agent
                            [rel_property_name] =>
                        )

                )

        )

)
```

Query Formation:

```
SELECT DISTINCT *  WHERE {
       {?Book a <http://dbpedia.org/ontology/Book>}
       {?Person a <http://xmlns.com/foaf/0.1/Person>}
       {?Person <http://xmlns.com/foaf/0.1/surname> ?Person_surname}
       {?Person <http://xmlns.com/foaf/0.1/givenName> ?Person_givenName}
       {?Book <http://dbpedia.org/ontology/author> ?Person}
       {?Book <http://xmlns.com/foaf/0.1/name> ?Book_name}
       {?Book <http://dbpedia.org/ontology/thumbnail> ?Book_thumbnail}
       {?Person <http://xmlns.com/foaf/0.1/name> ?Person_name}
FILTER(LANG(?Person_surname) = "en" && (REGEX(str(?Person_surname), "simon","i") || REGEX(str(?Person_givenName), "simon","i")))
}
```

Example 2

Query 2:

"list authors of physics"

Syntax Analysis:

```
Array
(
    [2] => Array
        (
            [1] => simon
            [2] => books
        )

)
```

Figure 8: Query 2 - Syntax Analysis

Concept Mapping:

```
Classes Found
NULL
Properties Found
Array
(
    [0] => Array
        (
            [type] => P
            [word] => authors
            [property] => http://purl.org/dc/terms/creator
        )

)
Instances Found
Array
(
    [0] => Array
        (
            [type] => I
            [word] => physics
            [class] => Array
                (
                    [0] =>  http://purl.org/dc/terms/BibliographicResource
                    [1] =>  http://purl.org/ontology/bibo/Book
                )

            [property] => http://purl.org/dc/terms/title
        )

    [1] => Array
        (
            [type] => I
            [word] => physics
            [class] => Array
                (
                    [0] =>  http://purl.org/ontology/bibo/Book
                    [1] =>  http://purl.org/dc/terms/BibliographicResource
                )

            [property] => http://purl.org/dc/terms/title
        )

)
```

Query Formation:

```
SELECT ?Book ?Book_title WHERE {
   {?Book <http://purl.org/dc/terms/creator> ?Person}
   {?Book  a <http://purl.org/ontology/bibo/Book>}
   {?Book <http://purl.org/dc/terms/title> ?Book_title}
   FILTER(REGEX(str(?Book_title),"physics","i"))
}
```

Example 3

Query 3:

"search for a book with ISBN 9780415435864"

Syntax Analysis:

```
Array
(
    [7] => Array
        (
            [type] => nummod
            [0] => ISBN
            [1] => 9780415435864
        )

    [4] => Array
        (
            [1] => search
            [case] => for
            [4] => book
        )

    [6] => Array
        (
            [4] => book
            [case] => with
            [6] => ISBN
        )

)
```

Figure 9: Query 3 - Syntax Analysis

Concept Mapping:

```
Classes Found
Array
(
    [0] => Array
        (
            [type] => C
            [word] => book
            [class] => http://purl.org/ontology/bibo/Book
        )

)
Properties Found
Array
(
    [0] => Array
        (
            [type] => P
            [word] => isbn
            [property] => http://purl.org/ontology/bibo/isbn13
        )

)
Instances Found
Array
(
    [0] => Array
        (
            [type] => I
            [word] => book
            [class] => Array
                (
                    [0] =>  http://purl.org/dc/terms/BibliographicResource
                    [1] =>  http://purl.org/ontology/bibo/Series
                )

            [property] => http://www.w3.org/2000/01/rdf-schema#label
        )

    [1] => Array
        (
            [type] => I
            [word] => 9780415435864
            [class] => Array
                (
                    [0] => http://purl.org/ontology/bibo/Book
                )

            [property] => http://purl.org/ontology/bibo/isbn13
        )

)
```

Query Formation:

```
SELECT DISTINCT * WHERE {
    {? Book a <http :// purl .org / ontology / bibo /Book >}
    {? Book <http :// purl .org / ontology / bibo /isbn13 > âĂŸ9780415435864 âĂŹ}
    {? Book <http :// purl .org /dc/ terms /title > ? Book_title }
}
```

Example 4

Query 4:

"Find 50 authors born in 1945"

Syntax Analysis:

```
Array
(
    [4] => Array
        (
            [type] => acl
            [0] => authors
            [1] => born
        )

    [1] => Array
        (
            [type] => root
            [0] => List authors
        )

    [6] => Array
        (
            [4] => born
            [case] => in
            [6] => 1945
        )

)
```

Figure 10: Query 4 - Syntax Analysis

Concept Mapping:

```
Classes Found
Null
Properties Found
Array
(
    [0] => Array
        (
            [type] => P
            [word] => authors
            [property] => http://purl.org/dc/terms/creator
        )

)
Instances Found
Array
(
    [0] => Array
        (
            [type] => I
            [word] => 1945
            [class] => Array
                (
                    [0] =>  http://purl.org/vocab/bio/0.1/Birth
                )

            [property] => http://www.w3.org/2000/01/rdf-schema#label
        )
```

```
    [1] => Array
        (
            [type] => I
            [word] => 1945
            [class] => Array
                (
                    [0] =>  http://purl.org/vocab/bio/0.1/Death
                )

            [property] => http://www.w3.org/2000/01/rdf-schema#label
        )

)
```

Query Formation:

```
SELECT DISTINCT * WHERE {
    {?var1 <http://purl.org/dc/terms/creator> ?Person}
    {?Person <http://purl.org/vocab/bio/0.1/event> ?Event}
    {?Event a <http://purl.org/vocab/bio/0.1/Birth>}
    {?Event <http://purl.org/vocab/bio/0.1/date> ?Event_date}
    {?Person <http://xmlns.com/foaf/0.1/name> ?Person_name}
FILTER (?Event_date = "1945"^^<http://www.w3.org/2001/XMLSchema#gYear>)
}LIMIT 50
```

Example 5

Query 5:

"titles by detective writer Ian Rankin"

Syntax Analysis:

```
Array
(
    [4] => Array
        (
            [1] => titles
            [case] => by
            [4] => Ian Rankin
        )

    [9] => Array
        (
            [4] => Rankin
            [9] => writer
        )

)
```

Figure 11: Query 5 - Syntax Analysis

Concept Mapping:

```
Classes Found
Null
Properties Found
Array
(
    [0] => Array
        (
            [type] => P
            [word] => titles
            [property] => http://dbpedia.org/ontology/title
        )

    [1] => Array
        (
            [type] => P
            [word] => writer
```

```
            [property] => http://purl.org/dc/terms/creator
        )

)
Instances Found
Array
(
    [0] => Array
        (
            [type] => I
            [word] => Ian
            [class] => Array
                (
                    [0] =>  http://purl.org/dc/terms/Agent
                    [1] =>  http://xmlns.com/foaf/0.1/Agent
                    [2] =>  http://xmlns.com/foaf/0.1/Person
                )

            [property] => http://xmlns.com/foaf/0.1/givenName
        )

    [1] => Array
        (
            [type] => I
            [word] => Ian
            [class] => Array
                (
                    [0] =>  http://purl.org/dc/terms/Agent
                    [1] =>  http://xmlns.com/foaf/0.1/Person
                    [2] =>  http://xmlns.com/foaf/0.1/Agent
                )

            [property] => http://xmlns.com/foaf/0.1/givenName
        )

    [2] => Array
        (
            [type] => I
            [word] => Ian
            [class] => Array
                (
                    [0] =>  http://xmlns.com/foaf/0.1/Agent
                    [1] =>  http://xmlns.com/foaf/0.1/Person
                    [2] =>  http://purl.org/dc/terms/Agent
                )

            [property] => http://xmlns.com/foaf/0.1/givenName
        )

    [3] => Array
        (
            [type] => I
            [word] => Ian
            [class] => Array
                (
                    [0] =>  http://xmlns.com/foaf/0.1/Person
                    [1] =>  http://purl.org/dc/terms/Agent
                    [2] =>  http://xmlns.com/foaf/0.1/Agent
                )

            [property] => http://xmlns.com/foaf/0.1/givenName
        )

    [4] => Array
        (
            [type] => I
            [word] => Ian
            [class] => Array
                (
                    [0] =>  http://xmlns.com/foaf/0.1/Person
                    [1] =>  http://xmlns.com/foaf/0.1/Agent
                    [2] =>  http://purl.org/dc/terms/Agent
                )

            [property] => http://xmlns.com/foaf/0.1/givenName
        )

    [5] => Array
        (
            [type] => I
            [word] => Rankin
            [class] => Array
                (
                    [0] =>  http://purl.org/dc/terms/Agent
                    [1] =>  http://xmlns.com/foaf/0.1/Person
                    [2] =>  http://xmlns.com/foaf/0.1/Agent
                )

            [property] => http://xmlns.com/foaf/0.1/familyName
        )

)
```

Query Formation:

```
SELECT DISTINCT * WHERE {
    {?Book <http://purl.org/dc/terms/creator> ?Person}
    {?Person a <http://xmlns.com/foaf/0.1/Person>}
    {?Person <http://xmlns.com/foaf/0.1/name> "Ian Rankin"}
    {?Book <http://purl.org/dc/terms/title> ?Book_title}
}
```

Appendix VI - A Snapshot of Server Log for Ontology Processor

```
run:
log4j:WARN No appenders could be found for logger (org.apache.jena.riot.stream.JenaIOEnvironment).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Class Found :  http://dbpedia.org/ontology/Guitar--Guitar
Class Found :  http://dbpedia.org/ontology/Guitar--guitars
========Property--http://dbpedia.org/ontology/secondPopularVote-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/foundingDate-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/highestMountain-domain-null-range-http://dbpedia.org/ontology/Mountain-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/leader-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/statValue-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/governmentType-domain-null-range-http://dbpedia.org/ontology/GovernmentType-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/namedAfter-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/year-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/requirement-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/agglomerationPopulationTotal-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/pccSecretary-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/startOccupation-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/managerYearsStartYear-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/pubchem-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/dateOfAbandonment-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/area-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/type-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/formerName-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/percentage-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/picturesCommonsCategory-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/mouthElevation-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/min-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/colourName-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/partialFailedLaunches-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/isPartOf-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/date-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/city-domain-null-range-http://dbpedia.org/ontology/City-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/editor-domain-null-range-http://dbpedia.org/ontology/Agent-ns-http://dbpedia.org/ontology/
--In statement for----editor-- "http://dbpedia.org/ontology/editor"
------editor--Sub Properties http://dbpedia.org/ontology/editor--editor
========Property--http://dbpedia.org/ontology/homeStadium-domain-null-range-http://dbpedia.org/ontology/Stadium-ns-http://dbpedia.org/ontology/
--In statement for----homeStadium-- "http://dbpedia.org/ontology/homeStadium"
------homeStadium--Sub Properties http://dbpedia.org/ontology/homeStadium--homeStadium
========Property--http://dbpedia.org/ontology/genre-domain-null-range-http://dbpedia.org/ontology/Genre-ns-http://dbpedia.org/ontology/
--In statement for----genre-- "http://dbpedia.org/ontology/genre", "http://schema.org/genre"
========Property--http://schema.org/genre-domain-null-range-null-ns-http://schema.org/
------genre--Sub Properties http://dbpedia.org/ontology/literaryGenre--literaryGenre
------genre--Sub Properties http://dbpedia.org/ontology/genre--genre
========Property--http://dbpedia.org/ontology/chiefPlace-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/
--In statement for----chiefPlace-- "http://dbpedia.org/ontology/chiefPlace"
------chiefPlace--Sub Properties http://dbpedia.org/ontology/chiefPlace--chiefPlace
========Property--http://dbpedia.org/ontology/associatedBand-domain-null-range-http://dbpedia.org/ontology/Band-ns-http://dbpedia.org/ontology/
--In statement for----associatedBand-- "http://dbpedia.org/ontology/associatedBand"
------associatedBand--Sub Properties http://dbpedia.org/ontology/associatedBand--associatedBand
========Property--http://dbpedia.org/ontology/taoiseach-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----taoiseach-- "http://dbpedia.org/ontology/taoiseach"
------taoiseach--Sub Properties http://dbpedia.org/ontology/taoiseach--taoiseach
========Property--http://dbpedia.org/ontology/draftYear-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
--In statement for----draftYear-- "http://dbpedia.org/ontology/draftYear"
------draftYear--Sub Properties http://dbpedia.org/ontology/draftYear--draftYear
========Property--http://dbpedia.org/ontology/memberOfParliament-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----memberOfParliament-- "http://dbpedia.org/ontology/memberOfParliament"
------memberOfParliament--Sub Properties http://dbpedia.org/ontology/memberOfParliament--memberOfParliament
========Property--http://dbpedia.org/ontology/productionStartYear-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
--In statement for----productionStartYear-- "http://dbpedia.org/ontology/productionStartYear"
------productionStartYear--Sub Properties http://dbpedia.org/ontology/productionStartYear--productionStartYear
========Property--http://dbpedia.org/ontology/owningCompany-domain-null-range-http://dbpedia.org/ontology/Company-ns-http://dbpedia.org/ontology/
--In statement for----owningCompany-- "http://dbpedia.org/ontology/owningCompany"
------owningCompany--Sub Properties http://dbpedia.org/ontology/owningCompany--owningCompany
========Property--http://dbpedia.org/ontology/name-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
--In statement for----name-- "http://dbpedia.org/ontology/name"
------name--Sub Properties http://dbpedia.org/ontology/ngcName--ngcName
------name--Sub Properties http://dbpedia.org/ontology/messierName--messierName
------name--Sub Properties http://dbpedia.org/ontology/name--name
========Property--http://dbpedia.org/ontology/percentageOfAreaWater-domain-null-range-http://www.w3.org/2001/XMLSchema#float-ns-http://dbpedia.org/ontology/
--In statement for----percentageOfAreaWater-- "http://dbpedia.org/ontology/percentageOfAreaWater"
------percentageOfAreaWater--Sub Properties http://dbpedia.org/ontology/percentageOfAreaWater--percentageOfAreaWater
========Property--http://dbpedia.org/ontology/royalAnthem-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/partyNumber-domain-null-range-http://www.w3.org/2001/XMLSchema#integer-ns-http://dbpedia.org/ontology/
--In statement for----partyNumber-- "http://dbpedia.org/ontology/partyNumber"
------partyNumber--Sub Properties http://dbpedia.org/ontology/partyNumber--partyNumber
========Property--http://dbpedia.org/ontology/state-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/
--In statement for----state-- "http://dbpedia.org/ontology/state"
------state--Sub Properties http://dbpedia.org/ontology/state--state
========Property--http://dbpedia.org/ontology/einecsNumber-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----einecsNumber-- "http://dbpedia.org/ontology/einecsNumber"
------einecsNumber--Sub Properties http://dbpedia.org/ontology/einecsNumber--einecsNumber
```

=======Property--http://dbpedia.org/ontology/geology-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----geology-- "http://dbpedia.org/ontology/geology"
------geology--Sub Properties http://dbpedia.org/ontology/geology--geology
=======Property--http://dbpedia.org/ontology/setupTime-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----setupTime-- "http://dbpedia.org/ontology/setupTime"
------setupTime--Sub Properties http://dbpedia.org/ontology/setupTime--setupTime
=======Property--http://dbpedia.org/ontology/orbits-domain-null-range-http://www.w3.org/2001/XMLSchema#integer-ns-http://dbpedia.org/ontology/
--In statement for----orbits-- "http://dbpedia.org/ontology/orbits"
------orbits--Sub Properties http://dbpedia.org/ontology/orbits--orbits
=======Property--http://dbpedia.org/ontology/company-domain-null-range-http://dbpedia.org/ontology/Organisation-ns-http://dbpedia.org/ontology/
--In statement for----company-- "http://dbpedia.org/ontology/company"
------company--Sub Properties http://dbpedia.org/ontology/company--company
=======Property--http://dbpedia.org/ontology/firstPopularVote-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----firstPopularVote-- "http://dbpedia.org/ontology/firstPopularVote"
------firstPopularVote--Sub Properties http://dbpedia.org/ontology/firstPopularVote--firstPopularVote
=======Property--http://dbpedia.org/ontology/format-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasSetting-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/monarch-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----monarch-- "http://dbpedia.org/ontology/monarch"
------monarch--Sub Properties http://dbpedia.org/ontology/monarch--monarch
=======Property--http://dbpedia.org/ontology/militaryGovernment-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----militaryGovernment-- "http://dbpedia.org/ontology/militaryGovernment"
------militaryGovernment--Sub Properties http://dbpedia.org/ontology/militaryGovernment--militaryGovernment
=======Property--http://dbpedia.org/ontology/wikiPageHistoryLink-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/MemberOfParliament-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isLocationOf-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/totalLaunches-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----totalLaunches-- "http://dbpedia.org/ontology/totalLaunches"
------totalLaunches--Sub Properties http://dbpedia.org/ontology/totalLaunches--totalLaunches
=======Property--http://dbpedia.org/ontology/currentCity-domain-null-range-http://dbpedia.org/ontology/City-ns-http://dbpedia.org/ontology/
--In statement for----currentCity-- "http://dbpedia.org/ontology/currentCity"
------currentCity--Sub Properties http://dbpedia.org/ontology/currentCity--currentCity
=======Property--http://dbpedia.org/ontology/range-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----range-- "http://dbpedia.org/ontology/range"
------range--Sub Properties http://dbpedia.org/ontology/range--range
=======Property--http://dbpedia.org/ontology/vehicle-domain-null-range-http://dbpedia.org/ontology/Automobile-ns-http://dbpedia.org/ontology/
--In statement for----vehicle-- "http://dbpedia.org/ontology/vehicle"
------vehicle--Sub Properties http://dbpedia.org/ontology/vehicle--vehicle
=======Property--http://dbpedia.org/ontology/orogeny-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/updated-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----updated-- "http://dbpedia.org/ontology/updated"
------updated--Sub Properties http://dbpedia.org/ontology/updated--updated
=======Property--http://dbpedia.org/ontology/sales-domain-null-range-http://dbpedia.org/ontology/Sales-ns-http://dbpedia.org/ontology/
--In statement for----sales-- "http://dbpedia.org/ontology/sales"
------sales--Sub Properties http://dbpedia.org/ontology/sales--sales
=======Property--http://dbpedia.org/ontology/restoreDate-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----restoreDate-- "http://dbpedia.org/ontology/restoreDate"
------restoreDate--Sub Properties http://dbpedia.org/ontology/restoreDate--restoreDate
=======Property--http://dbpedia.org/ontology/systemRequirements-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----systemRequirements-- "http://dbpedia.org/ontology/systemRequirements"
------systemRequirements--Sub Properties http://dbpedia.org/ontology/systemRequirements--systemRequirements
=======Property--http://dbpedia.org/ontology/activeYearsEndYear-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
--In statement for----activeYearsEndYear-- "http://dbpedia.org/ontology/activeYearsEndYear"
------activeYearsEndYear--Sub Properties http://dbpedia.org/ontology/activeYearsEndYear--activeYearsEndYear
=======Property--http://dbpedia.org/ontology/related-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/stateDelegate-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----stateDelegate-- "http://dbpedia.org/ontology/stateDelegate"
------stateDelegate--Sub Properties http://dbpedia.org/ontology/stateDelegate--stateDelegate
=======Property--http://dbpedia.org/ontology/atcPrefix-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----atcPrefix-- "http://dbpedia.org/ontology/atcPrefix"
------atcPrefix--Sub Properties http://dbpedia.org/ontology/atcPrefix--atcPrefix
=======Property--http://dbpedia.org/ontology/secondPlace-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----secondPlace-- "http://dbpedia.org/ontology/secondPlace"
------secondPlace--Sub Properties http://dbpedia.org/ontology/secondPlace--secondPlace
=======Property--http://dbpedia.org/ontology/demonym-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
--In statement for----demonym-- "http://dbpedia.org/ontology/demonym"
------demonym--Sub Properties http://dbpedia.org/ontology/demonym--demonym
=======Property--http://dbpedia.org/ontology/league-domain-null-range-http://dbpedia.org/ontology/SportsLeague-ns-http://dbpedia.org/ontology/
--In statement for----league-- "http://dbpedia.org/ontology/league"
------league--Sub Properties http://dbpedia.org/ontology/league--league
--In statement for----classification-- "http://dbpedia.org/ontology/classification"
------classification--Sub Properties http://dbpedia.org/ontology/classification--classification
=======Property--http://dbpedia.org/ontology/album-domain-null-range-http://dbpedia.org/ontology/Album-ns-http://dbpedia.org/ontology/
--In statement for----album-- "http://dbpedia.org/ontology/album"
------album--Sub Properties http://dbpedia.org/ontology/album--album
=======Property--http://dbpedia.org/ontology/picture-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/temperature-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----temperature-- "http://dbpedia.org/ontology/temperature"
------temperature--Sub Properties http://dbpedia.org/ontology/temperature--temperature
=======Property--http://dbpedia.org/ontology/volume-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----volume-- "http://dbpedia.org/ontology/volume"
------volume--Sub Properties http://dbpedia.org/ontology/volume--volume
=======Property--http://dbpedia.org/ontology/wikiPageInDegree-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----wikiPageInDegree-- "http://dbpedia.org/ontology/wikiPageInDegree"
------wikiPageInDegree--Sub Properties http://dbpedia.org/ontology/wikiPageInDegree--wikiPageInDegree
=======Property--http://dbpedia.org/ontology/religion-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/uses-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/hasInput-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/meltingPoint-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----meltingPoint-- "http://dbpedia.org/ontology/meltingPoint"
------meltingPoint--Sub Properties http://dbpedia.org/ontology/meltingPoint--meltingPoint
=======Property--http://dbpedia.org/ontology/licensee-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----licensee-- "http://dbpedia.org/ontology/licensee"
------licensee--Sub Properties http://dbpedia.org/ontology/licensee--licensee
=======Property--http://dbpedia.org/ontology/callSign-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----callSign-- "http://dbpedia.org/ontology/callSign"
------callSign--Sub Properties http://dbpedia.org/ontology/callSign--callSign
=======Property--http://dbpedia.org/ontology/conflict-domain-null-range-http://dbpedia.org/ontology/MilitaryConflict-ns-http://dbpedia.org/ontology/
--In statement for----conflict-- "http://dbpedia.org/ontology/conflict"
------conflict--Sub Properties http://dbpedia.org/ontology/conflict--conflict
=======Property--http://dbpedia.org/ontology/connotation-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/speaker-domain-null-range-http://www.w3.org/2001/XMLSchema#integer-ns-http://dbpedia.org/ontology/
--In statement for----speaker-- "http://dbpedia.org/ontology/speaker"
------speaker--Sub Properties http://dbpedia.org/ontology/speaker--speaker
=======Property--http://dbpedia.org/ontology/retired-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----retired-- "http://dbpedia.org/ontology/retired"
------retired--Sub Properties http://dbpedia.org/ontology/retired--retired
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#overlaps-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/numberOfSportsEvents-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----numberOfSportsEvents-- "http://dbpedia.org/ontology/numberOfSportsEvents"
------numberOfSportsEvents--Sub Properties http://dbpedia.org/ontology/numberOfSportsEvents--numberOfSportsEvents
=======Property--http://dbpedia.org/ontology/viceChancellor-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----viceChancellor-- "http://dbpedia.org/ontology/viceChancellor"
------viceChancellor--Sub Properties http://dbpedia.org/ontology/viceChancellor--viceChancellor
=======Property--http://dbpedia.org/ontology/sourceDistrict-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/

```
--In statement for----sourceDistrict-- "http://dbpedia.org/ontology/sourceDistrict"
------sourceDistrict--Sub Properties http://dbpedia.org/ontology/sourceDistrict--sourceDistrict
========Property--http://dbpedia.org/ontology/average-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----average-- "http://dbpedia.org/ontology/average"
------average--Sub Properties http://dbpedia.org/ontology/average--average
========Property--http://dbpedia.org/ontology/electionMajority-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----electionMajority-- "http://dbpedia.org/ontology/electionMajority"
------electionMajority--Sub Properties http://dbpedia.org/ontology/electionMajority--electionMajority
========Property--http://dbpedia.org/ontology/Wins-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/musicians-domain-http://dbpedia.org/ontology/Instrument-range-http://dbpedia.org/ontology/MusicalArtist-ns-http://dbpedia.org/ontology/
--In statement for----musicians-- "http://dbpedia.org/ontology/musicians"
------musicians--Sub Properties http://dbpedia.org/ontology/musicians--musicians
========Property--http://dbpedia.org/ontology/mayor-domain-null-range-http://dbpedia.org/ontology/Mayor-ns-http://dbpedia.org/ontology/
--In statement for----mayor-- "http://dbpedia.org/ontology/mayor"
------mayor--Sub Properties http://dbpedia.org/ontology/mayor--mayor
========Property--http://dbpedia.org/ontology/listItemOf-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/homeColourHexCode-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----homeColourHexCode-- "http://dbpedia.org/ontology/homeColourHexCode"
------homeColourHexCode--Sub Properties http://dbpedia.org/ontology/homeColourHexCode--homeColourHexCode
========Property--http://dbpedia.org/ontology/wikiPageRevisionID-domain-null-range-http://www.w3.org/2001/XMLSchema#integer-ns-http://dbpedia.org/ontology/
--In statement for----wikiPageRevisionID-- "http://dbpedia.org/ontology/wikiPageRevisionID"
------wikiPageRevisionID--Sub Properties http://dbpedia.org/ontology/wikiPageRevisionID--wikiPageRevisionID
========Property--http://dbpedia.org/ontology/organisation-domain-null-range-http://dbpedia.org/ontology/Organisation-ns-http://dbpedia.org/ontology/
--In statement for----organisation-- "http://dbpedia.org/ontology/organisation"
------organisation--Sub Properties http://dbpedia.org/ontology/organisation--organisation
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isMemberOf-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/vicePrimeMinister-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----vicePrimeMinister-- "http://dbpedia.org/ontology/vicePrimeMinister"
------vicePrimeMinister--Sub Properties http://dbpedia.org/ontology/vicePrimeMinister--vicePrimeMinister
========Property--http://dbpedia.org/ontology/management-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/floodingDate-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----floodingDate-- "http://dbpedia.org/ontology/floodingDate"
------floodingDate--Sub Properties http://dbpedia.org/ontology/floodingDate--floodingDate
========Property--http://dbpedia.org/ontology/imageSize-domain-null-range-http://www.w3.org/2001/XMLSchema#integer-ns-http://dbpedia.org/ontology/
--In statement for----imageSize-- "http://dbpedia.org/ontology/imageSize"
------imageSize--Sub Properties http://dbpedia.org/ontology/imageSize--imageSize
========Property--http://dbpedia.org/ontology/abbreviation-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----abbreviation-- "http://dbpedia.org/ontology/abbreviation"
------abbreviation--Sub Properties http://dbpedia.org/ontology/abbreviation--abbreviation
========Property--http://dbpedia.org/ontology/surfaceFormOccurrenceOffset-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----surfaceFormOccurrenceOffset-- "http://dbpedia.org/ontology/surfaceFormOccurrenceOffset"
------surfaceFormOccurrenceOffset--Sub Properties http://dbpedia.org/ontology/surfaceFormOccurrenceOffset--surfaceFormOccurrenceOffset
========Property--http://dbpedia.org/ontology/existence-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/occupation-domain-null-range-http://dbpedia.org/ontology/PersonFunction-ns-http://dbpedia.org/ontology/
--In statement for----occupation-- "http://dbpedia.org/ontology/occupation", "http://wikidata.dbpedia.org/resource/P106"
========Property--http://wikidata.dbpedia.org/resource/P106-domain-null-range-null-ns-http://wikidata.dbpedia.org/resource/
------occupation--Sub Properties http://dbpedia.org/ontology/occupation--occupation
========Property--http://dbpedia.org/ontology/thumbnail-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/mascot-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----mascot-- "http://dbpedia.org/ontology/mascot"
------mascot--Sub Properties http://dbpedia.org/ontology/mascot--mascot
========Property--http://dbpedia.org/ontology/logo-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----logo-- "http://dbpedia.org/ontology/logo"
------logo--Sub Properties http://dbpedia.org/ontology/logo--logo
========Property--http://dbpedia.org/ontology/tradeMark-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/race-domain-null-range-http://dbpedia.org/ontology/Race-ns-http://dbpedia.org/ontology/
--In statement for----race-- "http://dbpedia.org/ontology/race"
------race--Sub Properties http://dbpedia.org/ontology/race--race
========Property--http://dbpedia.org/ontology/averageSpeed-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----averageSpeed-- "http://dbpedia.org/ontology/averageSpeed"
------averageSpeed--Sub Properties http://dbpedia.org/ontology/averageSpeed--averageSpeed
========Property--http://dbpedia.org/ontology/activeYearsStartDate-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----activeYearsStartDate-- "http://dbpedia.org/ontology/activeYearsStartDate"
------activeYearsStartDate--Sub Properties http://dbpedia.org/ontology/activeYearsStartDate--activeYearsStartDate
========Property--http://dbpedia.org/ontology/publiclyAccessible-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----publiclyAccessible-- "http://dbpedia.org/ontology/publiclyAccessible"
------publiclyAccessible--Sub Properties http://dbpedia.org/ontology/publiclyAccessible--publiclyAccessible
========Property--http://dbpedia.org/ontology/chancellor-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----chancellor-- "http://dbpedia.org/ontology/chancellor"
------chancellor--Sub Properties http://dbpedia.org/ontology/chancellor--chancellor
========Property--http://dbpedia.org/ontology/series-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/foundingYear-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
--In statement for----foundingYear-- "http://dbpedia.org/ontology/foundingYear"
------foundingYear--Sub Properties http://dbpedia.org/ontology/foundingYear--foundingYear
========Property--http://dbpedia.org/ontology/country-domain-null-range-http://dbpedia.org/ontology/Country-ns-http://dbpedia.org/ontology/
--In statement for----country-- "http://dbpedia.org/ontology/country", "http://wikidata.dbpedia.org/resource/P17"
========Property--http://wikidata.dbpedia.org/resource/P17-domain-null-range-null-ns-http://wikidata.dbpedia.org/resource/
------country--Sub Properties http://dbpedia.org/ontology/country--country
========Property--http://dbpedia.org/ontology/publicationDate-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----publicationDate-- "http://dbpedia.org/ontology/publicationDate"
------publicationDate--Sub Properties http://dbpedia.org/ontology/publicationDate--publicationDate
========Property--http://dbpedia.org/ontology/alias-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----alias-- "http://dbpedia.org/ontology/alias"
------alias--Sub Properties http://dbpedia.org/ontology/alias--alias
========Property--http://dbpedia.org/ontology/LanguageCode-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/seniority-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----seniority-- "http://dbpedia.org/ontology/seniority"
------seniority--Sub Properties http://dbpedia.org/ontology/seniority--seniority
========Property--http://dbpedia.org/ontology/notes-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----notes-- "http://dbpedia.org/ontology/notes"
------notes--Sub Properties http://dbpedia.org/ontology/notes--notes
========Property--http://dbpedia.org/ontology/division-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/language-domain-null-range-http://dbpedia.org/ontology/Language-ns-http://dbpedia.org/ontology/
--In statement for----language-- "http://dbpedia.org/ontology/language", "http://schema.org/inLanguage"
========Property--http://schema.org/inLanguage-domain-null-range-null-ns-http://schema.org/
------language--Sub Properties http://dbpedia.org/ontology/originalLanguage--originalLanguage
------language--Sub Properties http://dbpedia.org/ontology/language--language
========Property--http://dbpedia.org/ontology/access-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----access-- "http://dbpedia.org/ontology/access"
------access--Sub Properties http://dbpedia.org/ontology/access--access
========Property--http://dbpedia.org/ontology/nameAsOf-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
--In statement for----nameAsOf-- "http://dbpedia.org/ontology/nameAsOf"
------nameAsOf--Sub Properties http://dbpedia.org/ontology/nameAsOf--nameAsOf
========Property--http://dbpedia.org/ontology/show-domain-null-range-http://dbpedia.org/ontology/TelevisionShow-ns-http://dbpedia.org/ontology/
--In statement for----show-- "http://dbpedia.org/ontology/show"
------show--Sub Properties http://dbpedia.org/ontology/show--show
========Property--http://dbpedia.org/ontology/minimumDischarge-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----minimumDischarge-- "http://dbpedia.org/ontology/minimumDischarge"
------minimumDischarge--Sub Properties http://dbpedia.org/ontology/minimumDischarge--minimumDischarge
========Property--http://dbpedia.org/ontology/longName-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
--In statement for----longName-- "http://dbpedia.org/ontology/longName"
------longName--Sub Properties http://dbpedia.org/ontology/longName--longName
========Property--http://dbpedia.org/ontology/lowestPlace-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/
--In statement for----lowestPlace-- "http://dbpedia.org/ontology/lowestPlace"
------lowestPlace--Sub Properties http://dbpedia.org/ontology/lowestPlace--lowestPlace
```

=======Property--http://dbpedia.org/ontology/number-domain-null-range-http://www.w3.org/2001/XMLSchema#integer-ns-http://dbpedia.org/ontology/
--In statement for----number-- "http://dbpedia.org/ontology/number"
------number--Sub Properties http://dbpedia.org/ontology/number--number
=======Property--http://dbpedia.org/ontology/office-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----office-- "http://dbpedia.org/ontology/office"
------office--Sub Properties http://dbpedia.org/ontology/office--office
=======Property--http://dbpedia.org/ontology/autonomy-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/sourcePosition-domain-null-range-http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing-ns-http://dbpedia.org/ontology/
--In statement for----sourcePosition-- "http://dbpedia.org/ontology/sourcePosition"
------sourcePosition--Sub Properties http://dbpedia.org/ontology/sourcePosition--sourcePosition
=======Property--http://dbpedia.org/ontology/managementPosition-domain-null-range-http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing-ns-http://dbpedia.org/ontology/
--In statement for----managementPosition-- "http://dbpedia.org/ontology/managementPosition"
------managementPosition--Sub Properties http://dbpedia.org/ontology/managementPosition--managementPosition
=======Property--http://dbpedia.org/ontology/wikiPageInterLanguageLink-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/aitaCode-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----aitaCode-- "http://dbpedia.org/ontology/aitaCode"
------aitaCode--Sub Properties http://dbpedia.org/ontology/aitaCode--aitaCode
=======Property--http://dbpedia.org/ontology/channel-domain-null-range-http://dbpedia.org/ontology/Broadcaster-ns-http://dbpedia.org/ontology/
--In statement for----channel-- "http://dbpedia.org/ontology/channel"
------channel--Sub Properties http://dbpedia.org/ontology/channel--channel
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isSpecializedBy-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/vicePresident-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----vicePresident-- "http://dbpedia.org/ontology/vicePresident"
------vicePresident--Sub Properties http://dbpedia.org/ontology/vicePresident--vicePresident
=======Property--http://dbpedia.org/ontology/authority-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/documentDesignation-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----documentDesignation-- "http://dbpedia.org/ontology/documentDesignation"
------documentDesignation--Sub Properties http://dbpedia.org/ontology/documentDesignation--documentDesignation
=======Property--http://dbpedia.org/ontology/currency-domain-null-range-http://dbpedia.org/ontology/Currency-ns-http://dbpedia.org/ontology/
--In statement for----currency-- "http://dbpedia.org/ontology/currency"
------currency--Sub Properties http://dbpedia.org/ontology/currency--currency
=======Property--http://dbpedia.org/ontology/hasVariant-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/timeInSpace-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----timeInSpace-- "http://dbpedia.org/ontology/timeInSpace"
------timeInSpace--Sub Properties http://dbpedia.org/ontology/timeInSpace--timeInSpace
=======Property--http://dbpedia.org/ontology/wikiPageWikiLink-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/animal-domain-null-range-http://dbpedia.org/ontology/Animal-ns-http://dbpedia.org/ontology/
--In statement for----animal-- "http://dbpedia.org/ontology/animal"
------animal--Sub Properties http://dbpedia.org/ontology/animal--animal
=======Property--http://dbpedia.org/ontology/chairman-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----chairman-- "http://dbpedia.org/ontology/chairman"
------chairman--Sub Properties http://dbpedia.org/ontology/chairman--chairman
=======Property--http://dbpedia.org/ontology/sourcePlace-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/
--In statement for----sourcePlace-- "http://dbpedia.org/ontology/sourcePlace"
------sourcePlace--Sub Properties http://dbpedia.org/ontology/sourcePlace--sourcePlace
=======Property--http://dbpedia.org/ontology/description-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
--In statement for----description-- "http://dbpedia.org/ontology/description"
------description--Sub Properties http://dbpedia.org/ontology/description--description
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isRoleOf-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasRegion-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/administrativeStatus-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----administrativeStatus-- "http://dbpedia.org/ontology/administrativeStatus"
------administrativeStatus--Sub Properties http://dbpedia.org/ontology/administrativeStatus--administrativeStatus
=======Property--http://dbpedia.org/ontology/value-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----value-- "http://dbpedia.org/ontology/value"
------value--Sub Properties http://dbpedia.org/ontology/value--value
=======Property--http://dbpedia.org/ontology/currentStatus-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----currentStatus-- "http://dbpedia.org/ontology/currentStatus"
------currentStatus--Sub Properties http://dbpedia.org/ontology/currentStatus--currentStatus
=======Property--http://dbpedia.org/ontology/numberOfVisitors-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----numberOfVisitors-- "http://dbpedia.org/ontology/numberOfVisitors"
------numberOfVisitors--Sub Properties http://dbpedia.org/ontology/numberOfVisitors--numberOfVisitors
=======Property--http://dbpedia.org/ontology/releaseYear-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/pastMember-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----pastMember-- "http://dbpedia.org/ontology/pastMember"
------pastMember--Sub Properties http://dbpedia.org/ontology/pastMember--pastMember
=======Property--http://dbpedia.org/ontology/sessionNumber-domain-null-range-http://www.w3.org/2001/XMLSchema#integer-ns-http://dbpedia.org/ontology/
--In statement for----sessionNumber-- "http://dbpedia.org/ontology/sessionNumber"
------sessionNumber--Sub Properties http://dbpedia.org/ontology/sessionNumber--sessionNumber
=======Property--http://dbpedia.org/ontology/flagLink-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----flagLink-- "http://dbpedia.org/ontology/flagLink"
------flagLink--Sub Properties http://dbpedia.org/ontology/flagLink--flagLink
=======Property--http://dbpedia.org/ontology/density-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----density-- "http://dbpedia.org/ontology/density"
------density--Sub Properties http://dbpedia.org/ontology/density--density
=======Property--http://dbpedia.org/ontology/mayorCouncillor-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----mayorCouncillor-- "http://dbpedia.org/ontology/mayorCouncillor"
------mayorCouncillor--Sub Properties http://dbpedia.org/ontology/mayorCouncillor--mayorCouncillor
=======Property--http://dbpedia.org/ontology/wikiPageModified-domain-null-range-http://www.w3.org/2001/XMLSchema#dateTime-ns-http://dbpedia.org/ontology/
--In statement for----wikiPageModified-- "http://dbpedia.org/ontology/wikiPageModified"
------wikiPageModified--Sub Properties http://dbpedia.org/ontology/wikiPageModified--wikiPageModified
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasParticipant-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/successor-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasRole-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/owner-domain-null-range-http://dbpedia.org/ontology/Agent-ns-http://dbpedia.org/ontology/
--In statement for----owner-- "http://dbpedia.org/ontology/owner"
------owner--Sub Properties http://dbpedia.org/ontology/owner--owner
=======Property--http://dbpedia.org/ontology/engineer-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----engineer-- "http://dbpedia.org/ontology/engineer"
------engineer--Sub Properties http://dbpedia.org/ontology/engineer--engineer
=======Property--http://dbpedia.org/ontology/follows-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/wikiPageRevisionLink-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/forces-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----forces-- "http://dbpedia.org/ontology/forces"
------forces--Sub Properties http://dbpedia.org/ontology/forces--forces
=======Property--http://dbpedia.org/ontology/locationCountry-domain-null-range-http://dbpedia.org/ontology/Country-ns-http://dbpedia.org/ontology/
--In statement for----locationCountry-- "http://dbpedia.org/ontology/locationCountry"
------locationCountry--Sub Properties http://dbpedia.org/ontology/locationCountry--locationCountry
=======Property--http://dbpedia.org/ontology/associate-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----associate-- "http://dbpedia.org/ontology/associate"
------associate--Sub Properties http://dbpedia.org/ontology/associate--associate
=======Property--http://dbpedia.org/ontology/region-domain-null-range-http://dbpedia.org/ontology/Place-ns-http://dbpedia.org/ontology/
--In statement for----region-- "http://dbpedia.org/ontology/region"
------region--Sub Properties http://dbpedia.org/ontology/region--region
=======Property--http://dbpedia.org/ontology/part-domain-null-range-http://dbpedia.org/ontology/Place-ns-http://dbpedia.org/ontology/
--In statement for----part-- "http://dbpedia.org/ontology/part"
------part--Sub Properties http://dbpedia.org/ontology/part--part
=======Property--http://dbpedia.org/ontology/chairLabel-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/televisionSeries-domain-null-range-http://dbpedia.org/ontology/TelevisionShow-ns-http://dbpedia.org/ontology/
--In statement for----televisionSeries-- "http://dbpedia.org/ontology/televisionSeries"
------televisionSeries--Sub Properties http://dbpedia.org/ontology/televisionSeries--televisionSeries
=======Property--http://dbpedia.org/ontology/lieutenant-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----lieutenant-- "http://dbpedia.org/ontology/lieutenant"
------lieutenant--Sub Properties http://dbpedia.org/ontology/lieutenant--lieutenant
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#precedes-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#

=======Property--http://dbpedia.org/ontology/translatedMotto-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----translatedMotto-- "http://dbpedia.org/ontology/translatedMotto"
------translatedMotto--Sub Properties http://dbpedia.org/ontology/translatedMotto--translatedMotto
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isAbout-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/elementAbove-domain-null-range-http://dbpedia.org/ontology/ChemicalSubstance-ns-http://dbpedia.org/ontology/
--In statement for----elementAbove-- "http://dbpedia.org/ontology/elementAbove"
------elementAbove--Sub Properties http://dbpedia.org/ontology/elementAbove--elementAbove
=======Property--http://dbpedia.org/ontology/predecessor-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/location-domain-null-range-http://dbpedia.org/ontology/Place-ns-http://dbpedia.org/ontology/
--In statement for----location-- "http://dbpedia.org/ontology/location"
------location--Sub Properties http://dbpedia.org/ontology/locationCountry--locationCountry
------location--Sub Properties http://dbpedia.org/ontology/locationCity--locationCity
------location--Sub Properties http://dbpedia.org/ontology/location--location
=======Property--http://dbpedia.org/ontology/frequency-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----frequency-- "http://dbpedia.org/ontology/frequency"
------frequency--Sub Properties http://dbpedia.org/ontology/frequency--frequency
=======Property--http://dbpedia.org/ontology/ColourName-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/foundedBy-domain-null-range-http://dbpedia.org/ontology/Agent-ns-http://dbpedia.org/ontology/
--In statement for----foundedBy-- "http://dbpedia.org/ontology/foundedBy", "http://schema.org/founders"
=======Property--http://schema.org/founders-domain-null-range-null-ns-http://schema.org/
------foundedBy--Sub Properties http://dbpedia.org/ontology/foundedBy--foundedBy
=======Property--http://dbpedia.org/ontology/flagBorder-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----flagBorder-- "http://dbpedia.org/ontology/flagBorder"
------flagBorder--Sub Properties http://dbpedia.org/ontology/flagBorder--flagBorder
=======Property--http://dbpedia.org/ontology/sport-domain-null-range-http://dbpedia.org/ontology/Sport-ns-http://dbpedia.org/ontology/
--In statement for----sport-- "http://dbpedia.org/ontology/sport"
------sport--Sub Properties http://dbpedia.org/ontology/sport--sport
=======Property--http://dbpedia.org/ontology/numberOfSports-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----numberOfSports-- "http://dbpedia.org/ontology/numberOfSports"
------numberOfSports--Sub Properties http://dbpedia.org/ontology/numberOfSports--numberOfSports
=======Property--http://dbpedia.org/ontology/flag-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----flag-- "http://dbpedia.org/ontology/flag"
------flag--Sub Properties http://dbpedia.org/ontology/flag--flag
=======Property--http://dbpedia.org/ontology/committee-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----committee-- "http://dbpedia.org/ontology/committee"
------committee--Sub Properties http://dbpedia.org/ontology/committee--committee
=======Property--http://dbpedia.org/ontology/pole-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----pole-- "http://dbpedia.org/ontology/pole"
------pole--Sub Properties http://dbpedia.org/ontology/pole--pole
=======Property--http://dbpedia.org/ontology/distributor-domain-null-range-http://dbpedia.org/ontology/Organisation-ns-http://dbpedia.org/ontology/
--In statement for----distributor-- "http://dbpedia.org/ontology/distributor"
------distributor--Sub Properties http://dbpedia.org/ontology/distributor--distributor
=======Property--http://dbpedia.org/ontology/kinOfLanguage-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----kinOfLanguage-- "http://dbpedia.org/ontology/kinOfLanguage"
------kinOfLanguage--Sub Properties http://dbpedia.org/ontology/kinOfLanguage--kinOfLanguage
=======Property--http://dbpedia.org/ontology/height-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----height-- "http://dbpedia.org/ontology/height"
------height--Sub Properties http://dbpedia.org/ontology/height--height
=======Property--http://dbpedia.org/ontology/raceHorse-domain-null-range-http://dbpedia.org/ontology/RaceHorse-ns-http://dbpedia.org/ontology/
--In statement for----raceHorse-- "http://dbpedia.org/ontology/raceHorse"
------raceHorse--Sub Properties http://dbpedia.org/ontology/raceHorse--raceHorse
=======Property--http://dbpedia.org/ontology/wikiPageEditLink-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasMember-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#specializes-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/activeYearsStartYear-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
--In statement for----activeYearsStartYear-- "http://dbpedia.org/ontology/activeYearsStartYear"
------activeYearsStartYear--Sub Properties http://dbpedia.org/ontology/activeYearsStartYear--activeYearsStartYear
=======Property--http://dbpedia.org/ontology/filename-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----filename-- "http://dbpedia.org/ontology/filename"
------filename--Sub Properties http://dbpedia.org/ontology/filename--filename
=======Property--http://dbpedia.org/ontology/deFactoLanguage-domain-null-range-http://dbpedia.org/ontology/Language-ns-http://dbpedia.org/ontology/
--In statement for----deFactoLanguage-- "http://dbpedia.org/ontology/deFactoLanguage", "http://dbpedia.org/ontology/language"
------deFactoLanguage--Sub Properties http://dbpedia.org/ontology/deFactoLanguage--deFactoLanguage
=======Property--http://dbpedia.org/ontology/budget-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----budget-- "http://dbpedia.org/ontology/budget"
------budget--Sub Properties http://dbpedia.org/ontology/budget--budget
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#coparticipatesWith-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isParticipantIn-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/designer-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----designer-- "http://dbpedia.org/ontology/designer"
------designer--Sub Properties http://dbpedia.org/ontology/designer--designer
=======Property--http://dbpedia.org/ontology/comment-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----comment-- "http://dbpedia.org/ontology/comment"
------comment--Sub Properties http://dbpedia.org/ontology/comment--comment
=======Property--http://dbpedia.org/ontology/purpose-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----purpose-- "http://dbpedia.org/ontology/purpose"
------purpose--Sub Properties http://dbpedia.org/ontology/purpose--purpose
=======Property--http://dbpedia.org/ontology/colour-domain-null-range-http://dbpedia.org/ontology/Colour-ns-http://dbpedia.org/ontology/
--In statement for----colour-- "http://dbpedia.org/ontology/colour"
------colour--Sub Properties http://dbpedia.org/ontology/colour--colour
=======Property--http://dbpedia.org/ontology/pronunciation-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----pronunciation-- "http://dbpedia.org/ontology/pronunciation"
------pronunciation--Sub Properties http://dbpedia.org/ontology/pronunciation--pronunciation
=======Property--http://dbpedia.org/ontology/draft-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----draft-- "http://dbpedia.org/ontology/draft"
------draft--Sub Properties http://dbpedia.org/ontology/draft--draft
=======Property--http://dbpedia.org/ontology/localAuthority-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/appointer-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----appointer-- "http://dbpedia.org/ontology/appointer"
------appointer--Sub Properties http://dbpedia.org/ontology/appointer--appointer
=======Property--http://dbpedia.org/ontology/synonym-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----synonym-- "http://dbpedia.org/ontology/synonym"
------synonym--Sub Properties http://dbpedia.org/ontology/synonym--synonym
=======Property--http://dbpedia.org/ontology/managementPlace-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/
--In statement for----managementPlace-- "http://dbpedia.org/ontology/managementPlace"
------managementPlace--Sub Properties http://dbpedia.org/ontology/managementPlace--managementPlace
=======Property--http://dbpedia.org/ontology/failedLaunches-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----failedLaunches-- "http://dbpedia.org/ontology/failedLaunches"
------failedLaunches--Sub Properties http://dbpedia.org/ontology/failedLaunches--failedLaunches
=======Property--http://dbpedia.org/ontology/firstOwner-domain-null-range-http://dbpedia.org/ontology/Agent-ns-http://dbpedia.org/ontology/
--In statement for----firstOwner-- "http://dbpedia.org/ontology/firstOwner"
------firstOwner--Sub Properties http://dbpedia.org/ontology/firstOwner--firstOwner
=======Property--http://dbpedia.org/ontology/managementElevation-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----managementElevation-- "http://dbpedia.org/ontology/managementElevation"
------managementElevation--Sub Properties http://dbpedia.org/ontology/managementElevation--managementElevation
=======Property--http://dbpedia.org/ontology/produces-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/blazonRatio-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----blazonRatio-- "http://dbpedia.org/ontology/blazonRatio"
------blazonRatio--Sub Properties http://dbpedia.org/ontology/blazonRatio--blazonRatio
=======Property--http://dbpedia.org/ontology/recordLabel-domain-null-range-http://dbpedia.org/ontology/RecordLabel-ns-http://dbpedia.org/ontology/
--In statement for----recordLabel-- "http://dbpedia.org/ontology/recordLabel", "http://wikidata.dbpedia.org/resource/P264"
=======Property--http://wikidata.dbpedia.org/resource/P264-domain-null-range-null-ns-http://wikidata.dbpedia.org/resource/
------recordLabel--Sub Properties http://dbpedia.org/ontology/recordLabel--recordLabel
=======Property--http://dbpedia.org/ontology/runningMate-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----runningMate-- "http://dbpedia.org/ontology/runningMate"

------runningMate--Sub Properties http://dbpedia.org/ontology/runningMate--runningMate
========Property--http://dbpedia.org/ontology/productionStartDate-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----productionStartDate-- "http://dbpedia.org/ontology/productionStartDate"
------productionStartDate--Sub Properties http://dbpedia.org/ontology/productionStartDate--productionStartDate
========Property--http://dbpedia.org/ontology/wikiPageID-domain-null-range-http://www.w3.org/2001/XMLSchema#integer-ns-http://dbpedia.org/ontology/
--In statement for----wikiPageID-- "http://dbpedia.org/ontology/wikiPageID"
------wikiPageID--Sub Properties http://dbpedia.org/ontology/wikiPageID--wikiPageID
========Property--http://dbpedia.org/ontology/mandate-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----mandate-- "http://dbpedia.org/ontology/mandate"
------mandate--Sub Properties http://dbpedia.org/ontology/mandate--mandate
========Property--http://dbpedia.org/ontology/code-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----code-- "http://dbpedia.org/ontology/code"
------code--Sub Properties http://dbpedia.org/ontology/codeStockExchange--codeStockExchange
------code--Sub Properties http://dbpedia.org/ontology/codeProvincialMonument--codeProvincialMonument
------code--Sub Properties http://dbpedia.org/ontology/codeNationalMonument--codeNationalMonument
------code--Sub Properties http://dbpedia.org/ontology/codeMunicipalMonument--codeMunicipalMonument
------code--Sub Properties http://dbpedia.org/ontology/code--code
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasCommonBoundary-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/incumbent-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----incumbent-- "http://dbpedia.org/ontology/incumbent"
------incumbent--Sub Properties http://dbpedia.org/ontology/incumbent--incumbent
========Property--http://dbpedia.org/ontology/militaryUnitSize-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----militaryUnitSize-- "http://dbpedia.org/ontology/militaryUnitSize"
------militaryUnitSize--Sub Properties http://dbpedia.org/ontology/militaryUnitSize--militaryUnitSize
========Property--http://dbpedia.org/ontology/wikiPageCharacterSize-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----wikiPageCharacterSize-- "http://dbpedia.org/ontology/wikiPageCharacterSize"
------wikiPageCharacterSize--Sub Properties http://dbpedia.org/ontology/wikiPageCharacterSize--wikiPageCharacterSize
========Property--http://dbpedia.org/ontology/tennisSurfaceType-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----tennisSurfaceType-- "http://dbpedia.org/ontology/tennisSurfaceType"
------tennisSurfaceType--Sub Properties http://dbpedia.org/ontology/tennisSurfaceType--tennisSurfaceType
========Property--http://dbpedia.org/ontology/wikiPageDisambiguates-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/width-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----width-- "http://dbpedia.org/ontology/width"
------width--Sub Properties http://dbpedia.org/ontology/width--width
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasQuality-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/highschool-domain-null-range-http://dbpedia.org/ontology/School-ns-http://dbpedia.org/ontology/
--In statement for----highschool-- "http://dbpedia.org/ontology/highschool"
------highschool--Sub Properties http://dbpedia.org/ontology/highschool--highschool
========Property--http://dbpedia.org/ontology/team-domain-null-range-http://dbpedia.org/ontology/SportsTeam-ns-http://dbpedia.org/ontology/
--In statement for----team-- "http://dbpedia.org/ontology/team", "http://wikidata.dbpedia.org/resource/P54"
========Property--http://wikidata.dbpedia.org/resource/P54-domain-null-range-null-ns-http://wikidata.dbpedia.org/resource/
------team--Sub Properties http://dbpedia.org/ontology/team--team
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/playingTime-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----playingTime-- "http://dbpedia.org/ontology/playingTime"
------playingTime--Sub Properties http://dbpedia.org/ontology/playingTime--playingTime
========Property--http://dbpedia.org/ontology/time-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----time-- "http://dbpedia.org/ontology/time"
------time--Sub Properties http://dbpedia.org/ontology/time--time
========Property--http://dbpedia.org/ontology/gameModus-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----gameModus-- "http://dbpedia.org/ontology/gameModus"
------gameModus--Sub Properties http://dbpedia.org/ontology/gameModus--gameModus
========Property--http://dbpedia.org/ontology/thumbnailCaption-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----thumbnailCaption-- "http://dbpedia.org/ontology/thumbnailCaption"
------thumbnailCaption--Sub Properties http://dbpedia.org/ontology/thumbnailCaption--thumbnailCaption
========Property--http://dbpedia.org/ontology/motto-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----motto-- "http://dbpedia.org/ontology/motto"
------motto--Sub Properties http://dbpedia.org/ontology/motto--motto
========Property--http://dbpedia.org/ontology/rebuildDate-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----rebuildDate-- "http://dbpedia.org/ontology/rebuildDate"
------rebuildDate--Sub Properties http://dbpedia.org/ontology/rebuildDate--rebuildDate
========Property--http://dbpedia.org/ontology/crest-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/Type-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/creator-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----creator-- "http://dbpedia.org/ontology/creator"
------creator--Sub Properties http://dbpedia.org/ontology/creator--creator
========Property--http://dbpedia.org/ontology/event-domain-null-range-http://dbpedia.org/ontology/Event-ns-http://dbpedia.org/ontology/
--In statement for----event-- "http://dbpedia.org/ontology/event"
------event--Sub Properties http://dbpedia.org/ontology/event--event
========Property--http://dbpedia.org/ontology/highestPlace-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/
--In statement for----highestPlace-- "http://dbpedia.org/ontology/highestPlace"
------highestPlace--Sub Properties http://dbpedia.org/ontology/highestPlace--highestPlace
========Property--http://dbpedia.org/ontology/biome-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/portfolio-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----portfolio-- "http://dbpedia.org/ontology/portfolio"
------portfolio--Sub Properties http://dbpedia.org/ontology/portfolio--portfolio
========Property--http://dbpedia.org/ontology/length-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----length-- "http://dbpedia.org/ontology/length"
------length--Sub Properties http://dbpedia.org/ontology/length--length
========Property--http://dbpedia.org/ontology/variantOf-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/sourceState-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/
--In statement for----sourceState-- "http://dbpedia.org/ontology/sourceState"
------sourceState--Sub Properties http://dbpedia.org/ontology/sourceState--sourceState
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#sameSettingAs-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/apcPresident-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----apcPresident-- "http://dbpedia.org/ontology/apcPresident"
------apcPresident--Sub Properties http://dbpedia.org/ontology/apcPresident--apcPresident
========Property--http://dbpedia.org/ontology/closingDate-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----closingDate-- "http://dbpedia.org/ontology/closingDate"
------closingDate--Sub Properties http://dbpedia.org/ontology/closingDate--closingDate
========Property--http://dbpedia.org/ontology/followedBy-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/title-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
--In statement for----title-- "http://dbpedia.org/ontology/title"
------title--Sub Properties http://dbpedia.org/ontology/title--title
========Property--http://dbpedia.org/ontology/cosparId-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----cosparId-- "http://dbpedia.org/ontology/cosparId"
------cosparId--Sub Properties http://dbpedia.org/ontology/cosparId--cosparId
========Property--http://dbpedia.org/ontology/ethnicGroupsInYear-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
--In statement for----ethnicGroupsInYear-- "http://dbpedia.org/ontology/ethnicGroupsInYear"
------ethnicGroupsInYear--Sub Properties http://dbpedia.org/ontology/ethnicGroupsInYear--ethnicGroupsInYear
========Property--http://dbpedia.org/ontology/blazonLink-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----blazonLink-- "http://dbpedia.org/ontology/blazonLink"
------blazonLink--Sub Properties http://dbpedia.org/ontology/blazonLink--blazonLink
========Property--http://dbpedia.org/ontology/founder-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/drugbank-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----drugbank-- "http://dbpedia.org/ontology/drugbank"
------drugbank--Sub Properties http://dbpedia.org/ontology/drugbank--drugbank
========Property--http://dbpedia.org/ontology/siren-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----siren-- "http://dbpedia.org/ontology/siren"
------siren--Sub Properties http://dbpedia.org/ontology/siren--siren
========Property--http://dbpedia.org/ontology/background-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----background-- "http://dbpedia.org/ontology/background"
------background--Sub Properties http://dbpedia.org/ontology/background--background
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isClassifiedBy-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/managerYearsEndYear-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/

--In statement for----managerYearsEndYear-- "http://dbpedia.org/ontology/managerYearsEndYear"
------managerYearsEndYear--Sub Properties http://dbpedia.org/ontology/managerYearsEndYear--managerYearsEndYear
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasConstituent-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/Status-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/quotation-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----quotation-- "http://dbpedia.org/ontology/quotation"
------quotation--Sub Properties http://dbpedia.org/ontology/quotation--quotation
========Property--http://dbpedia.org/ontology/fuel-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/fdaUniiCode-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----fdaUniiCode-- "http://dbpedia.org/ontology/fdaUniiCode"
------fdaUniiCode--Sub Properties http://dbpedia.org/ontology/fdaUniiCode--fdaUniiCode
========Property--http://dbpedia.org/ontology/stadium-domain-null-range-http://dbpedia.org/ontology/Stadium-ns-http://dbpedia.org/ontology/
--In statement for----stadium-- "http://dbpedia.org/ontology/stadium"
------stadium--Sub Properties http://dbpedia.org/ontology/stadium--stadium
========Property--http://dbpedia.org/ontology/abstract-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
--In statement for----abstract-- "http://dbpedia.org/ontology/abstract"
------abstract--Sub Properties http://dbpedia.org/ontology/abstract--abstract
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isDescribedBy-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/thirdPlace-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----thirdPlace-- "http://dbpedia.org/ontology/thirdPlace"
------thirdPlace--Sub Properties http://dbpedia.org/ontology/thirdPlace--thirdPlace
========Property--http://dbpedia.org/ontology/productionEndYear-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
--In statement for----productionEndYear-- "http://dbpedia.org/ontology/productionEndYear"
------productionEndYear--Sub Properties http://dbpedia.org/ontology/productionEndYear--productionEndYear
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#conceptualizes-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/alongside-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----alongside-- "http://dbpedia.org/ontology/alongside"
------alongside--Sub Properties http://dbpedia.org/ontology/alongside--alongside
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#associatedWith-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/sourceElevation-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----sourceElevation-- "http://dbpedia.org/ontology/sourceElevation"
------sourceElevation--Sub Properties http://dbpedia.org/ontology/sourceElevation--sourceElevation
========Property--http://dbpedia.org/ontology/areaMetro-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----areaMetro-- "http://dbpedia.org/ontology/areaMetro"
------areaMetro--Sub Properties http://dbpedia.org/ontology/areaMetro--areaMetro
========Property--http://dbpedia.org/ontology/usk-domain-null-range-http://www.w3.org/2001/XMLSchema#integer-ns-http://dbpedia.org/ontology/
--In statement for----usk-- "http://dbpedia.org/ontology/usk"
------usk--Sub Properties http://dbpedia.org/ontology/usk--usk
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isPartOf-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/imageFlag-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----imageFlag-- "http://dbpedia.org/ontology/imageFlag", "http://wikidata.dbpedia.org/resource/P41"
========Property--http://wikidata.dbpedia.org/resource/P41-domain-null-range-null-ns-http://wikidata.dbpedia.org/resource/
------imageFlag--Sub Properties http://dbpedia.org/ontology/imageFlag--imageFlag
========Property--http://dbpedia.org/ontology/sizeMap-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----sizeMap-- "http://dbpedia.org/ontology/sizeMap"
------sizeMap--Sub Properties http://dbpedia.org/ontology/sizeMap--sizeMap
========Property--http://dbpedia.org/ontology/atcSupplemental-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----atcSupplemental-- "http://dbpedia.org/ontology/atcSupplemental"
------atcSupplemental--Sub Properties http://dbpedia.org/ontology/atcSupplemental--atcSupplemental
========Property--http://dbpedia.org/ontology/websiteLabel-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
--In statement for----websiteLabel-- "http://dbpedia.org/ontology/websiteLabel"
------websiteLabel--Sub Properties http://dbpedia.org/ontology/websiteLabel--websiteLabel
========Property--http://dbpedia.org/ontology/provides-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/orderInOffice-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----orderInOffice-- "http://dbpedia.org/ontology/orderInOffice"
------orderInOffice--Sub Properties http://dbpedia.org/ontology/orderInOffice--orderInOffice
========Property--http://dbpedia.org/ontology/topSpeed-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----topSpeed-- "http://dbpedia.org/ontology/topSpeed"
------topSpeed--Sub Properties http://dbpedia.org/ontology/topSpeed--topSpeed
========Property--http://dbpedia.org/ontology/depictionDescription-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
--In statement for----depictionDescription-- "http://dbpedia.org/ontology/depictionDescription"
------depictionDescription--Sub Properties http://dbpedia.org/ontology/depictionDescription--depictionDescription
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#follows-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/fileSize-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----fileSize-- "http://dbpedia.org/ontology/fileSize"
------fileSize--Sub Properties http://dbpedia.org/ontology/fileSize--fileSize
========Property--http://dbpedia.org/ontology/commandant-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----commandant-- "http://dbpedia.org/ontology/commandant"
------commandant--Sub Properties http://dbpedia.org/ontology/commandant--commandant
========Property--http://dbpedia.org/ontology/Code-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/Distance-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/distance-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----distance-- "http://dbpedia.org/ontology/distance"
------distance--Sub Properties http://dbpedia.org/ontology/distance--distance
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#unifies-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/playRole-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/president-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----president-- "http://dbpedia.org/ontology/president"
------president--Sub Properties http://dbpedia.org/ontology/president--president
========Property--http://dbpedia.org/ontology/source-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/teamSize-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----teamSize-- "http://dbpedia.org/ontology/teamSize"
------teamSize--Sub Properties http://dbpedia.org/ontology/teamSize--teamSize
========Property--http://dbpedia.org/ontology/pictureDescription-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isSettingFor-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/endOccupation-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----endOccupation-- "http://dbpedia.org/ontology/endOccupation"
------endOccupation--Sub Properties http://dbpedia.org/ontology/endOccupation--endOccupation
========Property--http://dbpedia.org/ontology/coach-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----coach-- "http://dbpedia.org/ontology/coach"
------coach--Sub Properties http://dbpedia.org/ontology/coach--coach
========Property--http://dbpedia.org/ontology/wikiPageExternalLink-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/successfulLaunches-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----successfulLaunches-- "http://dbpedia.org/ontology/successfulLaunches"
------successfulLaunches--Sub Properties http://dbpedia.org/ontology/successfulLaunches--successfulLaunches
========Property--http://dbpedia.org/ontology/Tribus-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/wikiPageLength-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----wikiPageLength-- "http://dbpedia.org/ontology/wikiPageLength"
------wikiPageLength--Sub Properties http://dbpedia.org/ontology/wikiPageLength--wikiPageLength
========Property--http://dbpedia.org/ontology/mainInterest-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/japanName-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----japanName-- "http://dbpedia.org/ontology/japanName"
------japanName--Sub Properties http://dbpedia.org/ontology/japanName--japanName
========Property--http://dbpedia.org/ontology/weight-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----weight-- "http://dbpedia.org/ontology/weight"
------weight--Sub Properties http://dbpedia.org/ontology/weight--weight
========Property--http://dbpedia.org/ontology/frenchNickname-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----frenchNickname-- "http://dbpedia.org/ontology/frenchNickname"
------frenchNickname--Sub Properties http://dbpedia.org/ontology/frenchNickname--frenchNickname
========Property--http://dbpedia.org/ontology/primeMinister-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----primeMinister-- "http://dbpedia.org/ontology/primeMinister"
------primeMinister--Sub Properties http://dbpedia.org/ontology/primeMinister--primeMinister
========Property--http://dbpedia.org/ontology/commonName-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
--In statement for----commonName-- "http://dbpedia.org/ontology/commonName"
------commonName--Sub Properties http://dbpedia.org/ontology/commonName--commonName

=======Property--http://dbpedia.org/ontology/dynasty-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/builder-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/Medalist-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/month-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----month-- "http://dbpedia.org/ontology/month"
------month--Sub Properties http://dbpedia.org/ontology/month--month
=======Property--http://dbpedia.org/ontology/identifiedBy-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----identifiedBy-- "http://dbpedia.org/ontology/identifiedBy"
------identifiedBy--Sub Properties http://dbpedia.org/ontology/hasNationalArchivesIdentifier--hasNationalArchivesIdentifier
------identifiedBy--Sub Properties http://dbpedia.org/ontology/documentNumber--documentNumber
------identifiedBy--Sub Properties http://dbpedia.org/ontology/identifiedBy--identifiedBy
=======Property--http://dbpedia.org/ontology/commander-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----commander-- "http://dbpedia.org/ontology/commander"
------commander--Sub Properties http://dbpedia.org/ontology/commander--commander
=======Property--http://dbpedia.org/ontology/productionEndDate-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----productionEndDate-- "http://dbpedia.org/ontology/productionEndDate"
------productionEndDate--Sub Properties http://dbpedia.org/ontology/productionEndDate--productionEndDate
=======Property--http://dbpedia.org/ontology/languageRegulator-domain-null-range-http://dbpedia.org/ontology/Language-ns-http://dbpedia.org/ontology/
--In statement for----languageRegulator-- "http://dbpedia.org/ontology/languageRegulator"
------languageRegulator--Sub Properties http://dbpedia.org/ontology/languageRegulator--languageRegulator
=======Property--http://dbpedia.org/ontology/sourceRegion-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/
--In statement for----sourceRegion-- "http://dbpedia.org/ontology/sourceRegion"
------sourceRegion--Sub Properties http://dbpedia.org/ontology/sourceRegion--sourceRegion
=======Property--http://dbpedia.org/ontology/blazon-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----blazon-- "http://dbpedia.org/ontology/blazon"
------blazon--Sub Properties http://dbpedia.org/ontology/blazon--blazon
=======Property--http://dbpedia.org/ontology/awayColourHexCode-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----awayColourHexCode-- "http://dbpedia.org/ontology/awayColourHexCode"
------awayColourHexCode--Sub Properties http://dbpedia.org/ontology/awayColourHexCode--awayColourHexCode
=======Property--http://dbpedia.org/ontology/originalName-domain-null-range-http://www.w3.org/1999/02/22-rdf-syntax-ns#langString-ns-http://dbpedia.org/ontology/
--In statement for----originalName-- "http://dbpedia.org/ontology/originalName"
------originalName--Sub Properties http://dbpedia.org/ontology/originalName--originalName
=======Property--http://dbpedia.org/ontology/manufacturer-domain-null-range-http://dbpedia.org/ontology/Organisation-ns-http://dbpedia.org/ontology/
--In statement for----manufacturer-- "http://dbpedia.org/ontology/manufacturer"
------manufacturer--Sub Properties http://dbpedia.org/ontology/manufacturer--manufacturer
=======Property--http://dbpedia.org/ontology/status-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----status-- "http://dbpedia.org/ontology/status"
------status--Sub Properties http://dbpedia.org/ontology/status--status
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasComponent-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/casSupplemental-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----casSupplemental-- "http://dbpedia.org/ontology/casSupplemental"
------casSupplemental--Sub Properties http://dbpedia.org/ontology/casSupplemental--casSupplemental
=======Property--http://dbpedia.org/ontology/battle-domain-null-range-http://dbpedia.org/ontology/MilitaryConflict-ns-http://dbpedia.org/ontology/
--In statement for----battle-- "http://dbpedia.org/ontology/battle"
------battle--Sub Properties http://dbpedia.org/ontology/battle--battle
=======Property--http://dbpedia.org/ontology/sizeLogo-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----sizeLogo-- "http://dbpedia.org/ontology/sizeLogo"
------sizeLogo--Sub Properties http://dbpedia.org/ontology/sizeLogo--sizeLogo
=======Property--http://dbpedia.org/ontology/termPeriod-domain-null-range-http://dbpedia.org/ontology/TimePeriod-ns-http://dbpedia.org/ontology/
--In statement for----termPeriod-- "http://dbpedia.org/ontology/termPeriod"
------termPeriod--Sub Properties http://dbpedia.org/ontology/termPeriod--termPeriod
=======Property--http://dbpedia.org/ontology/collectionSize-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/owningOrganisation-domain-null-range-http://dbpedia.org/ontology/Organisation-ns-http://dbpedia.org/ontology/
--In statement for----owningOrganisation-- "http://dbpedia.org/ontology/owningOrganisation"
------owningOrganisation--Sub Properties http://dbpedia.org/ontology/owningOrganisation--owningOrganisation
=======Property--http://dbpedia.org/ontology/municipality-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/
--In statement for----municipality-- "http://dbpedia.org/ontology/municipality"
------municipality--Sub Properties http://dbpedia.org/ontology/municipality--municipality
=======Property--http://dbpedia.org/ontology/destructionDate-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----destructionDate-- "http://dbpedia.org/ontology/destructionDate"
------destructionDate--Sub Properties http://dbpedia.org/ontology/destructionDate--destructionDate
=======Property--http://dbpedia.org/ontology/category-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/role-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----role-- "http://dbpedia.org/ontology/role"
------role--Sub Properties http://dbpedia.org/ontology/role--role
=======Property--http://dbpedia.org/ontology/jureLanguage-domain-null-range-http://dbpedia.org/ontology/Language-ns-http://dbpedia.org/ontology/
--In statement for----jureLanguage-- "http://dbpedia.org/ontology/jureLanguage", "http://dbpedia.org/ontology/language"
------jureLanguage--Sub Properties http://dbpedia.org/ontology/jureLanguage--jureLanguage
=======Property--http://dbpedia.org/ontology/period-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/individualisedGnd-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----individualisedGnd-- "http://dbpedia.org/ontology/individualisedGnd", "http://wikidata.dbpedia.org/resource/P227"
=======Property--http://wikidata.dbpedia.org/resource/P227-domain-null-range-null-ns-http://wikidata.dbpedia.org/resource/
------individualisedGnd--Sub Properties http://dbpedia.org/ontology/individualisedGnd--individualisedGnd
=======Property--http://dbpedia.org/ontology/nominee-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----nominee-- "http://dbpedia.org/ontology/nominee"
------nominee--Sub Properties http://dbpedia.org/ontology/nominee--nominee
=======Property--http://dbpedia.org/ontology/executiveProducer-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----executiveProducer-- "http://dbpedia.org/ontology/executiveProducer"
------executiveProducer--Sub Properties http://dbpedia.org/ontology/executiveProducer--executiveProducer
=======Property--http://dbpedia.org/ontology/firstLeader-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----firstLeader-- "http://dbpedia.org/ontology/firstLeader"
------firstLeader--Sub Properties http://dbpedia.org/ontology/firstLeader--firstLeader
=======Property--http://dbpedia.org/ontology/secondLeader-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----secondLeader-- "http://dbpedia.org/ontology/secondLeader"
------secondLeader--Sub Properties http://dbpedia.org/ontology/secondLeader--secondLeader
=======Property--http://dbpedia.org/ontology/senator-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----senator-- "http://dbpedia.org/ontology/senator"
------senator--Sub Properties http://dbpedia.org/ontology/senator--senator
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#nearTo-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/diameter-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----diameter-- "http://dbpedia.org/ontology/diameter"
------diameter--Sub Properties http://dbpedia.org/ontology/diameter--diameter
=======Property--http://dbpedia.org/ontology/lowestPosition-domain-null-range-http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing-ns-http://dbpedia.org/ontology/
--In statement for----lowestPosition-- "http://dbpedia.org/ontology/lowestPosition"
------lowestPosition--Sub Properties http://dbpedia.org/ontology/lowestPosition--lowestPosition
=======Property--http://dbpedia.org/ontology/associatedMusicalArtist-domain-null-range-http://dbpedia.org/ontology/MusicalArtist-ns-http://dbpedia.org/ontology/
--In statement for----associatedMusicalArtist-- "http://dbpedia.org/ontology/associatedMusicalArtist"
------associatedMusicalArtist--Sub Properties http://dbpedia.org/ontology/associatedMusicalArtist--associatedMusicalArtist
=======Property--http://dbpedia.org/ontology/associateStar-domain-null-range-http://dbpedia.org/ontology/Constellation-ns-http://dbpedia.org/ontology/
--In statement for----associateStar-- "http://dbpedia.org/ontology/associateStar"
------associateStar--Sub Properties http://dbpedia.org/ontology/associateStar--associateStar
=======Property--http://dbpedia.org/ontology/alternativeName-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----alternativeName-- "http://dbpedia.org/ontology/alternativeName"
------alternativeName--Sub Properties http://dbpedia.org/ontology/alias--alias
------alternativeName--Sub Properties http://dbpedia.org/ontology/alternativeName--alternativeName
=======Property--http://dbpedia.org/ontology/ascent-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----ascent-- "http://dbpedia.org/ontology/ascent"
------ascent--Sub Properties http://dbpedia.org/ontology/ascent--ascent
=======Property--http://dbpedia.org/ontology/discoverer-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----discoverer-- "http://dbpedia.org/ontology/discoverer", "http://wikidata.dbpedia.org/resource/P61"
=======Property--http://wikidata.dbpedia.org/resource/P61-domain-null-range-null-ns-http://wikidata.dbpedia.org/resource/
------discoverer--Sub Properties http://dbpedia.org/ontology/discoverer--discoverer
=======Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isExpressedBy-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
=======Property--http://dbpedia.org/ontology/skills-domain-null-range-null-ns-http://dbpedia.org/ontology/
=======Property--http://dbpedia.org/ontology/party-domain-null-range-http://dbpedia.org/ontology/PoliticalParty-ns-http://dbpedia.org/ontology/

```
--In statement for----party-- "http://dbpedia.org/ontology/party", "http://wikidata.dbpedia.org/resource/P102"
========Property--http://wikidata.dbpedia.org/resource/P102-domain-null-range-null-ns-http://wikidata.dbpedia.org/resource/
------party--Sub Properties http://dbpedia.org/ontology/party--party
========Property--http://dbpedia.org/ontology/parentCompany-domain-null-range-http://dbpedia.org/ontology/Company-ns-http://dbpedia.org/ontology/
--In statement for----parentCompany-- "http://dbpedia.org/ontology/parentCompany"
------parentCompany--Sub Properties http://dbpedia.org/ontology/parentCompany--parentCompany
========Property--http://dbpedia.org/ontology/homeArena-domain-null-range-http://dbpedia.org/ontology/Arena-ns-http://dbpedia.org/ontology/
--In statement for----homeArena-- "http://dbpedia.org/ontology/homeArena"
------homeArena--Sub Properties http://dbpedia.org/ontology/homeArena--homeArena
========Property--http://dbpedia.org/ontology/max-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----max-- "http://dbpedia.org/ontology/max"
------max--Sub Properties http://dbpedia.org/ontology/max--max
========Property--http://dbpedia.org/ontology/sizeBlazon-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----sizeBlazon-- "http://dbpedia.org/ontology/sizeBlazon"
------sizeBlazon--Sub Properties http://dbpedia.org/ontology/sizeBlazon--sizeBlazon
========Property--http://dbpedia.org/ontology/origin-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/
--In statement for----origin-- "http://dbpedia.org/ontology/origin"
------origin--Sub Properties http://dbpedia.org/ontology/origin--origin
========Property--http://dbpedia.org/ontology/gender-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/deliveryDate-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----deliveryDate-- "http://dbpedia.org/ontology/deliveryDate"
------deliveryDate--Sub Properties http://dbpedia.org/ontology/deliveryDate--deliveryDate
========Property--http://dbpedia.org/ontology/county-domain-null-range-http://dbpedia.org/ontology/PopulatedPlace-ns-http://dbpedia.org/ontology/
--In statement for----county-- "http://dbpedia.org/ontology/county"
------county--Sub Properties http://dbpedia.org/ontology/county--county
========Property--http://dbpedia.org/ontology/deputy-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----deputy-- "http://dbpedia.org/ontology/deputy"
------deputy--Sub Properties http://dbpedia.org/ontology/deputy--deputy
========Property--http://dbpedia.org/ontology/rating-domain-null-range-http://www.w3.org/2001/XMLSchema#float-ns-http://dbpedia.org/ontology/
--In statement for----rating-- "http://dbpedia.org/ontology/rating"
------rating--Sub Properties http://dbpedia.org/ontology/rating--rating
========Property--http://dbpedia.org/ontology/tag-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----tag-- "http://dbpedia.org/ontology/tag"
------tag--Sub Properties http://dbpedia.org/ontology/tag--tag
========Property--http://dbpedia.org/ontology/minorityFloorLeader-domain-null-range-http://www.w3.org/2001/XMLSchema#integer-ns-http://dbpedia.org/ontology/
--In statement for----minorityFloorLeader-- "http://dbpedia.org/ontology/minorityFloorLeader"
------minorityFloorLeader--Sub Properties http://dbpedia.org/ontology/minorityFloorLeader--minorityFloorLeader
========Property--http://dbpedia.org/ontology/webcast-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/maximumDischarge-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----maximumDischarge-- "http://dbpedia.org/ontology/maximumDischarge"
------maximumDischarge--Sub Properties http://dbpedia.org/ontology/maximumDischarge--maximumDischarge
========Property--http://dbpedia.org/ontology/position-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasPart-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/similar-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/prefix-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----prefix-- "http://dbpedia.org/ontology/prefix"
------prefix--Sub Properties http://dbpedia.org/ontology/prefix--prefix
========Property--http://dbpedia.org/ontology/atcSuffix-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----atcSuffix-- "http://dbpedia.org/ontology/atcSuffix"
------atcSuffix--Sub Properties http://dbpedia.org/ontology/atcSuffix--atcSuffix
========Property--http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#concretelyExpresses-domain-null-range-null-ns-http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
========Property--http://dbpedia.org/ontology/sizeThumbnail-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----sizeThumbnail-- "http://dbpedia.org/ontology/sizeThumbnail"
------sizeThumbnail--Sub Properties http://dbpedia.org/ontology/sizeThumbnail--sizeThumbnail
========Property--http://dbpedia.org/ontology/mass-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----mass-- "http://dbpedia.org/ontology/mass"
------mass--Sub Properties http://dbpedia.org/ontology/mass--mass
========Property--http://dbpedia.org/ontology/hasSurfaceForm-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----hasSurfaceForm-- "http://dbpedia.org/ontology/hasSurfaceForm"
------hasSurfaceForm--Sub Properties http://dbpedia.org/ontology/hasSurfaceForm--hasSurfaceForm
========Property--http://dbpedia.org/ontology/governor-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----governor-- "http://dbpedia.org/ontology/governor"
------governor--Sub Properties http://dbpedia.org/ontology/governor--governor
========Property--http://dbpedia.org/ontology/sportGoverningBody-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/populationTotal-domain-null-range-http://www.w3.org/2001/XMLSchema#nonNegativeInteger-ns-http://dbpedia.org/ontology/
--In statement for----populationTotal-- "http://dbpedia.org/ontology/populationTotal"
------populationTotal--Sub Properties http://dbpedia.org/ontology/populationTotal--populationTotal
========Property--http://dbpedia.org/ontology/sublimationPoint-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----sublimationPoint-- "http://dbpedia.org/ontology/sublimationPoint"
------sublimationPoint--Sub Properties http://dbpedia.org/ontology/sublimationPoint--sublimationPoint
========Property--http://dbpedia.org/ontology/hasChannel-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/sourceConfluenceElevation-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----sourceConfluenceElevation-- "http://dbpedia.org/ontology/sourceConfluenceElevation"
------sourceConfluenceElevation--Sub Properties http://dbpedia.org/ontology/sourceConfluenceElevation--sourceConfluenceElevation
========Property--http://dbpedia.org/ontology/majorityFloorLeader-domain-null-range-http://www.w3.org/2001/XMLSchema#integer-ns-http://dbpedia.org/ontology/
--In statement for----majorityFloorLeader-- "http://dbpedia.org/ontology/majorityFloorLeader"
------majorityFloorLeader--Sub Properties http://dbpedia.org/ontology/majorityFloorLeader--majorityFloorLeader
========Property--http://dbpedia.org/ontology/lowestMountain-domain-null-range-http://dbpedia.org/ontology/Mountain-ns-http://dbpedia.org/ontology/
--In statement for----lowestMountain-- "http://dbpedia.org/ontology/lowestMountain"
------lowestMountain--Sub Properties http://dbpedia.org/ontology/lowestMountain--lowestMountain
========Property--http://dbpedia.org/ontology/criteria-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----criteria-- "http://dbpedia.org/ontology/criteria"
------criteria--Sub Properties http://dbpedia.org/ontology/criteria--criteria
========Property--http://dbpedia.org/ontology/prominence-domain-null-range-http://www.w3.org/2001/XMLSchema#double-ns-http://dbpedia.org/ontology/
--In statement for----prominence-- "http://dbpedia.org/ontology/prominence"
------prominence--Sub Properties http://dbpedia.org/ontology/prominence--prominence
========Property--http://dbpedia.org/ontology/dateBudget-domain-null-range-http://www.w3.org/2001/XMLSchema#date-ns-http://dbpedia.org/ontology/
--In statement for----dateBudget-- "http://dbpedia.org/ontology/dateBudget"
------dateBudget--Sub Properties http://dbpedia.org/ontology/dateBudget--dateBudget
========Property--http://dbpedia.org/ontology/jurisdiction-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/rank-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----rank-- "http://dbpedia.org/ontology/rank"
------rank--Sub Properties http://dbpedia.org/ontology/rank--rank
========Property--http://dbpedia.org/ontology/sourceMountain-domain-null-range-http://dbpedia.org/ontology/Mountain-ns-http://dbpedia.org/ontology/
--In statement for----sourceMountain-- "http://dbpedia.org/ontology/sourceMountain"
------sourceMountain--Sub Properties http://dbpedia.org/ontology/sourceMountain--sourceMountain
========Property--http://dbpedia.org/ontology/endYear-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
--In statement for----endYear-- "http://dbpedia.org/ontology/endYear"
------endYear--Sub Properties http://dbpedia.org/ontology/endYear--endYear
========Property--http://dbpedia.org/ontology/startYear-domain-null-range-http://www.w3.org/2001/XMLSchema#gYear-ns-http://dbpedia.org/ontology/
--In statement for----startYear-- "http://dbpedia.org/ontology/startYear"
------startYear--Sub Properties http://dbpedia.org/ontology/startYear--startYear
========Property--http://dbpedia.org/ontology/wikiPageRedirects-domain-null-range-null-ns-http://dbpedia.org/ontology/
========Property--http://dbpedia.org/ontology/firstPlace-domain-null-range-http://www.w3.org/2001/XMLSchema#string-ns-http://dbpedia.org/ontology/
--In statement for----firstPlace-- "http://dbpedia.org/ontology/firstPlace"
------firstPlace--Sub Properties http://dbpedia.org/ontology/firstPlace--firstPlace
========Property--http://dbpedia.org/ontology/governorGeneral-domain-null-range-http://dbpedia.org/ontology/Person-ns-http://dbpedia.org/ontology/
--In statement for----governorGeneral-- "http://dbpedia.org/ontology/governorGeneral"
------governorGeneral--Sub Properties http://dbpedia.org/ontology/governorGeneral--governorGeneral
========Property--http://dbpedia.org/ontology/highestPosition-domain-null-range-http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing-ns-http://dbpedia.org/ontology/
--In statement for----highestPosition-- "http://dbpedia.org/ontology/highestPosition"
------highestPosition--Sub Properties http://dbpedia.org/ontology/highestPosition--highestPosition
```