



Universitetet  
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

## MASTER'S THESIS

Study program/specialization: Computer Science	Spring semester, 2017  Open
Author: Aida Mehdipour Pirbazari	... Aida Mehdipour..... (signature author)
Supervisor(s): Professor. Krisztian Balog	
Title of Master's Thesis: <b>Answering Engine for sports statistics</b> Subtitle: <b>Development of an ontology and a knowledge base</b>	
ECTS: 30	
Subject headings: Answering Engine, Ontology, Semantic Web, Ontological knowledge base, Linked Data Question Answering (QA)	Pages: ...78..... + attachments/other: 100  Stavanger, ...15/06/2017..... Date/year

UNIVERSITY OF STAVANGER

DEPARTMENT OF ELECTRICAL AND COMPUTER  
ENGINEERING

MASTER THESIS IN COMPUTER SCIENCE

---

# Answering Engine for Sports Statistics

Development of an ontology and a knowledge base

---

*Author:*  
Aida MEHDIPOUR  
PIRBAZARI

*Supervisor:*  
Prof. Krisztian BALOG

June 2017





---

## Preface

This work is carried out as a master thesis in computer science at the University of Stavanger during the Spring semester of 2017. The idea of this study, is one of the interesting topics in the field of Question Answering and semantic technologies which is suggested by the supervisor of the project. The objective is to build an answering engine that can interpret natural language questions related to statistics about Formula One and can respond appropriately. Due to relatively large work scale and time limitation, this study focuses on response generation part and my fellow student focuses on question interpretation part, making a prototype and evaluation tasks are considered as shared responsibilities.

Stavanger, Spring 2017



---

## Acknowledgements

I would first like to express my sincere gratitude to my thesis advisor and supervisor; Professor Krisztian Balog for his great and valuable comments, remarks, and engagement through the learning process of this master thesis. Furthermore, I would like to thank Øyvind Blaauw, my fellow student, for his perfect cooperation on the shared responsibilities.

Finally, I must express my profound gratitude to my family with great support and continuous encouragement during my studies and throughout the process of researching and writing this thesis.



---

## Abstract

Focus of this study is on development of a domain-specific ontology and a knowledge base of facts for a Question Answering system. This QA system accepts natural language questions about statistics of Formula One and transforms them in to formal queries using natural language processing techniques and the designed ontology. It then executes the queries against the knowledge base to return exact answers.

During the design process of the ontology, regular standards and regulations have been utilized, and the required data for implementing the ontology have been collected from a large-scale and reliable data source. Semantic technologies have been used to transform data to structured and machine-readable formats and a graph knowledge base is used for storage and retrieval of the structured data through formal queries.

The evaluation results show that the knowledge base covers lots of correct and relevant information about main entities in the given domain. The designed ontology has required potential to answer many statistical questions that it was designed for and the QA system based on this ontology can provide correct answers to easy questions about statistics of Formula One. The limitation of the ontology is that it cannot provide the QA system with the necessary knowledge to answer complex queries about statistics of Formula One.





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Thesis overview . . . . .	1
1.2	Contributions . . . . .	2
1.3	Thesis outline . . . . .	4
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Semantic QA systems . . . . .	5
2.2	Ontology . . . . .	7
2.2.1	Ontology definition . . . . .	8
2.2.2	Degree of formalization in an Ontology . . . . .	8
2.2.3	Ontology components . . . . .	8
2.2.4	Levels of an ontology . . . . .	9
2.2.5	Creation of an Ontology . . . . .	10
2.2.6	Ontology applications . . . . .	12
2.3	Semantic Web . . . . .	13
2.3.1	Definitions of the Semantic Web . . . . .	14
2.3.2	Architecture of the Semantic Web . . . . .	14
2.3.3	The Semantic Web technologies . . . . .	16
2.3.4	RDFS . . . . .	20
2.3.5	OWL . . . . .	20
2.3.6	SPARQL . . . . .	21
2.4	Linked Data . . . . .	22
2.4.1	Why Linked Data . . . . .	22
2.4.2	The principles of Linked Data . . . . .	23
2.4.3	Linking Open Data project . . . . .	23
2.4.4	Linked Data applications . . . . .	23
2.5	Knowledge-base QA systems . . . . .	24
<b>3</b>	<b>Development of the QA system</b>	<b>27</b>
3.1	Design of the ontology . . . . .	27
3.1.1	Specification . . . . .	28
3.1.2	Conceptualization . . . . .	33
3.2	Implementation of the ontology . . . . .	44
3.2.1	Collecting data . . . . .	44

---

3.2.2	Developing a knowledge base . . . . .	53
3.3	Extraction of answers . . . . .	56
3.4	Implementing a prototype . . . . .	56
<b>4</b>	<b>Evaluation</b>	<b>59</b>
4.1	Statistical analysis of the knowledge base . . . . .	59
4.2	Measuring the accuracy of the knowledge base . . . . .	63
4.2.1	Measurement of precision . . . . .	63
4.2.2	Measurement of recall . . . . .	65
4.3	Evaluating design of the ontology . . . . .	69
4.3.1	Baseline questions . . . . .	70
4.3.2	Analysis of errors . . . . .	72
<b>5</b>	<b>Conclusion</b>	<b>75</b>
5.1	Achievements . . . . .	75
5.2	Discussion . . . . .	76
5.3	Limitations . . . . .	77
5.4	Future work . . . . .	78
	<b>Appendices</b>	<b>79</b>

# List of Figures

1.1	Architecture of the QA system . . . . .	3
2.1	METHONTOLOGY life cycle [10] . . . . .	11
2.2	Ontological Question Answering Systems [7] . . . . .	13
2.3	Semantic Web Stack (Tim-Berners-Lee, 2015) . . . . .	15
2.4	A simple RDF graph describing the relationship between the race driver and the GP race . . . . .	17
2.5	A simple RDF graph with literal for describing data value . . . . .	18
2.6	A sample RDF document with N-Triples format . . . . .	19
2.7	A sample RDF/XML document . . . . .	20
2.8	An example of RDF/XML document using rdfs elements . . . . .	21
2.9	Linking Open Data cloud diagram [29] . . . . .	24
3.1	A sample list of competency questions and answers . . . . .	30
3.2	Formula One Reference Ontology Requirement Specifications Document . . . . .	32
3.3	Data dictionary in the domain of Formula One (based on DBpedia ontology) . . . . .	35
3.4	Table of Instances: Sebastian Vettel . . . . .	37
3.5	Preliminary conceptual model of Formula One domain based on DBpedia ontology . . . . .	38
3.6	Snapshot of a part of DBpedia ontology with focus on Formula One classes . . . . .	39
3.7	Snapshot of a part of DBpedia ontology with focus on Formula One instances . . . . .	40
3.8	Conceptual model of Formula One domain based on new ontology . . . . .	43
3.9	Snapshot of a DBpedia web page including schema triples about Grand Prix entity . . . . .	45
3.10	Snapshot of a Wikipedia web page including names, countries and total records of drivers . . . . .	46
3.11	Snapshot of a Wikipedia web page including current teams of drivers . . . . .	47
3.12	Snapshot of a Wikipedia web page including first and last wins of GP winners (drivers) . . . . .	47

---

3.13	Snapshot of a Wikipedia web page including records of drivers in season 2015 . . . . .	48
3.14	Snapshot of a Wikipedia web page including participant constructors and drivers in season 2014 . . . . .	49
3.15	Snapshot of a Wikipedia web page including general info. about season 2012 . . . . .	50
3.16	Snapshot of a Wikipedia web page including champions of all seasons . . . . .	51
3.17	Snapshot of a Wikipedia web page including total records of teams	52
3.18	Snapshot of a Wikipedia web page including first and last wins of GP winners (teams) . . . . .	52
3.19	Snapshot of a Dbpedia web page including sample triples about an instance of GP for extraction . . . . .	53
3.20	Process of making RDF triples and storing in a database . . . . .	54
3.21	Snapshot of GraphDB including schema triples . . . . .	55
3.22	Snapshot of GraphDB including triples of an instance of Grand prix . . . . .	55
3.23	User interface of the QA system while typing a question . . . . .	57
3.24	User interface of the QA system after hitting the Ask button . . . . .	58
4.1	Number of instances in F1 ontology . . . . .	60
4.2	Number of instances per season . . . . .	61
4.3	Percentage of defined predicates for instances of F1 classes . . . . .	62
4.4	Relationships between Classes in F1 ontology . . . . .	63
4.5	A sample list of baseline questions . . . . .	72

# List of Tables

2.1	Sample triple statements in RDF . . . . .	19
3.1	Sample terms related to Formula One racer . . . . .	31
3.2	Examples of objects . . . . .	31
3.3	Table of instance attribute: podiums . . . . .	35
3.4	Table of Instance attribute: distance Laps . . . . .	36
3.5	Table of Relation attribute: first Driver . . . . .	36
3.6	Table of Relation attribute: pole Team . . . . .	36
4.1	Number of instances in samples and the KB per class . . . . .	64
4.2	Detailed and total precision values of the samples from the KB . . . . .	65
4.3	Sample instances for recall evaluation . . . . .	66
4.4	Missing data found for sample instances . . . . .	68
4.5	Recall values for sample instances . . . . .	69
4.6	Detailed and total recall values for sample classes . . . . .	69
4.7	Categories of questions based on the answers . . . . .	71
4.8	Types of questions based on difficulty . . . . .	71
1	Contents of attachments . . . . .	80



# Chapter 1

## Introduction

This chapter consists of three sections. First section, introduces the problem statement that this thesis covers. Main contributions to solve the problem are presented in Section 1.2, as well as the thesis outline in Section 1.3.

### 1.1 Thesis overview

An answering engine is a question answering (QA) system which automatically answers questions posed by humans in natural language (NL) using either a pre-structured database or a collection of natural language documents. It attempts to deal with a wide range of question types including fact, list, definition, how, why, etc [33].

Unlike traditional search engines which are based on term matching techniques and retrieve all documents containing the keywords of user's query; QA systems as the next generation of search engines (semantic search engines), try to look for the search intent and contextual meaning of query words, to provide more precise and concrete answers (facts or text).

There are several techniques and technologies that semantic QA systems rely on, such as natural language processing (NLP) and machine learning algorithms to interpret the questions; ontologies to provide the system with any domain-specific conceptual knowledge and structured databases of knowledge to provide exact answers.

Among the semantic QA systems which is based on an ontology and a structured database, Wolfram Alpha is an excellent example of answering engine which introduces new methods for understanding linguistic inputs and is built purely entirely on semantic search technologies. Wolfram Alpha rather than crawl the web, uses semantic technologies on its own externally sourced "curated data" or structured data, that can be relied on by everyone for definitive answers to factual queries [38]. This engine shows the interpretation of the request along with a direct answer as the result of a query that is expressed in natural language format.



One issue with Wolfram Alpha is that its coverage is limited. It knows a lot about mathematics and even about general purpose (encyclopedia) knowledge, however it knows very little about sports. For example, if we submit a query about Formula One sport like *Which race driver won Australian Grand Prix in 2010?*; the search engine does not know about the statistics of the sport. It only simply returns a picture of Formula One races.

Motivated by all these facts and needs mentioned above, in this project, we propose an answering engine, based on a new ontology and semantic approach, that can interpret the natural language questions related to statistics of Formula One sport, extract answers from its pre-structured database and return direct answers to the user as well as underlying data that is used for the calculations.

## 1.2 Contributions

The project is presented by the University of Stavanger as a candidate assignment for a master thesis. The specific tasks that were specified for this project are listed below:

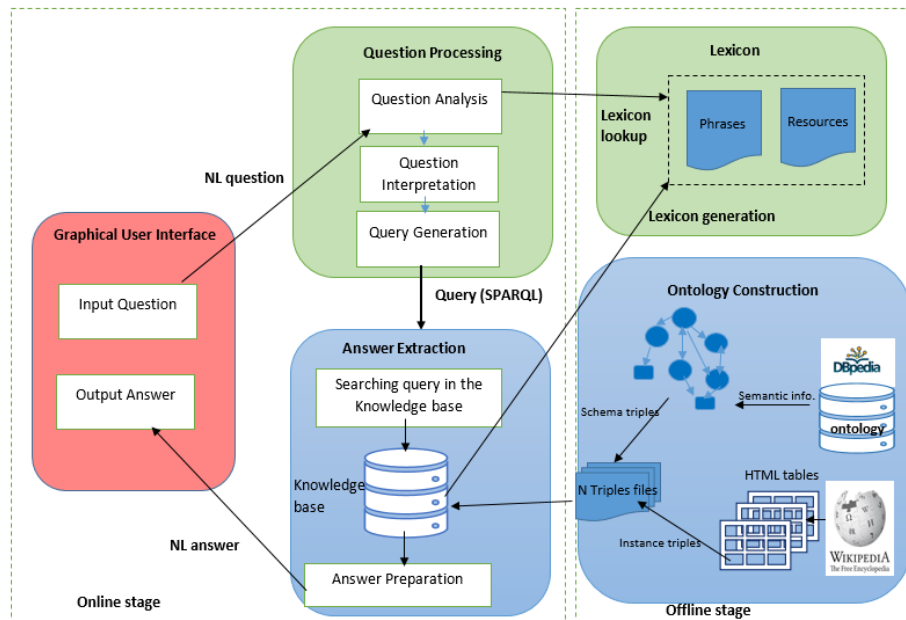
- Collecting data or connecting to existing data services.
- Building a conceptual model (ontology) of the chosen domain of sports.
- Developing a question interpreter that can understand concepts related to the selected field of sport (e.g., names of teams, players, leagues, years, etc.) and related to statistics (e.g., “best”, “most”, “highest”, “maximum”, “on average”, “during X”, etc.).
- Developing an answering module that can process the interpreted question against a knowledge base of facts. In addition to the factual answer, the system shows the underlying data that was used for the calculations (as “evidence”).
- Implementing a working prototype to generate a real version of the answering system, demonstrate, and test its performance.
- Performing an evaluation of the system to determine which aspects of the system are worthwhile and which parts are to be revised or ignored.

The project has been a cooperation between two students: “Øyvind Blaauw” and me. The thesis with Blaauw will focus on interpreting NL questions and converting them into queries that can be executed on the KB. The paper presented here, focuses on collecting data and developing a new ontology and a knowledge base. Building a prototype and performing evaluation are shared responsibilities. The evaluation presented in this study, focuses on ontology design and the accuracy of the knowledge base, whereas the evaluation presented by Blaauw, focuses on the performance of the “Question Processing” module.

Figure 1.1 shows the architecture of the proposed system including the modules which are responsible to perform the specified tasks. The modules *Answer*

*Extraction* and *Ontology Construction* colored by blue boxes in the figure, are contributions of this thesis and the modules *Question Processing* and *Lexicons*, colored by green boxes, are covered by Blaauw.

Figure 1.1: Architecture of the QA system



According to the model, the system works in two stages: offline and online. In online stage, after entering a natural language question by the user, using the module *GUI*, the *Question Processing* module analyzes and transforms the question into a formal query which is understandable for the machine. After the formal query was generated successfully, it is sent to the *Answer Extraction* module where the formal query is searched against the knowledge base and the answer is extracted and prepared in NL format and sent to the user through the user interface.

In contrast, offline stage shows which processes and data sources are used to support the modules of the system in online processing. In this stage, *Lexicons* module helps the system with interpreting the question and generating the query by generating semantic concepts driven from the ontology and the *Ontology Construction* module, is responsible for collecting data from data sources, developing an ontology and a knowledge base.

### 1.3 Thesis outline

The rest of this thesis is structured as follows. In Chapter 2, we review the background theories about semantic answering engines and all the related concepts including ontology, Semantic Web, Linked Data, and knowledge-based QA systems.

In Chapter 3, we describe the development process of the QA system, with focus on developing an ontology and a knowledge base for the system. We also explain how these resources are used in online stage to help the system to generate final answers. The last section, explains the implementation of a working prototype where we can run and test the first version of the QA system.

Evaluation of the QA system including experiments and results is provided in Chapter 4 and finally, in Chapter 5, we summarize our achievements and findings, outline the limitations and make improvement suggestions for future studies.

## Chapter 2

# Background

This chapter consists of five sections. First section, after listing limitations of traditional search paradigms, gives a short introduction to semantic search and its related concepts. Section 2.2, defines the concept of ontology and its related subjects. Sections 2.3 and 2.4 introduce Semantic Web as an extension of the current web, semantic technologies and Linked Data which makes the Semantic Web as a reality. Finally, QA systems based on the domain of questions and source of answers are studied in Section 2.5.

### 2.1 Semantic QA systems

Nowadays, in the World Wide Web, the huge amount of information and data repositories increase rapidly. This has motivated the need for efficient and powerful search strategies to move from organizing information to knowledge discovery. Classical search methods that search keywords of the query within documents and retrieve a ranked list of documents to the user, are based on content matching rather than meaning. If only taking advantage of keyword retrieval, there are many limitations as follows:

First, the search results are too comprehensive; typically, redundant, and irrelevant. The search engine cannot obtain and deliver the specific answer to the user for a specific question. There is still user who must spend time to search deeply in to a list of documents to find the appropriate answer.

Secondly, the results usually suffer from low level of reliability; huge amount of information is accessible through the Internet, mostly unstructured (in format of free text) where everyone can publish their ideas about different topics. This simply implies the possibilities for retrieving poor quality or unreliable information. Most search engines do not evaluate their sources nor asses the content of information they retrieve.

Moreover, there are no simple options for users to ask their questions in form of natural language, if the query is expressed in everyday language like the way in which a person is asking from an expert person, not a machine, mostly

the search results will not be precise enough. It means users are expected to formulate their queries in a couple of keywords that can be understandable for the search engine [27]. Furthermore, if the search domain is specialized, the searcher should be familiar with so much specialized terms in that domain.

Another important issue that Macgregor (2008) mentions is that two queries consisting similar keywords but having different meanings are not distinguishable for the search engine. He clarifies this issue with a simple example: “in the two queries *Books about recommender systems* versus *Systems that recommend books*, a mere keyword search would not suffice in distinguishing between the two queries. Consequently, similar results are retrieved despite the difference in the meaning between the two. While the first query requires a list of books about recommender systems, the second one requests information on a list of systems which recommend books. It is evident that additional information need to be taken into consideration to be able to effectively process such queries” [11].

Endeavors to overcome these limitations has led to significant improvements in search methods of some existing search engines. For instance, the powerful search engines like Google which process billions of searches queries each month, recently provides more explicit responses to common queries, as well as documents. For example, when querying Google for *Hotels in Stavanger*, the hotels’ names are displayed on the city map of Stavanger along with main information represented in tabular formats such as their prices and classes. When searching for a simple conversational query like *What’s the weather today?* weather information for the user’s current location as well as a 7-day prediction will automatically be returned . In many cases for more complex queries, however, Google falls back to the “10 blue links”.

It is obvious that “the difficulty of identifying and verifying the correct answer makes question answering more difficult than the common information retrieval task performed by search engines” [2]. Question Answering systems (QA systems) make use of essential technologies to address this issue. The automatic systems which can accept and interpret the natural language queries, locate, extract, and represent precise and meaningful answers to the users [3], rather providing list of documents or webpages. The new systems even provide sufficient content to validate the answers.

Today, various Question Answering systems have been developed that are different based on four dimensions: type of questions to accept (facts, dialogue, etc.), source of the answers (unstructured, semi structured, and structured data), the scope (domain specific or domain independent) and the manner of adaptability and disambiguation [1].

However, the common objective among new systems is that they can process natural language questions and look for the search intent and contextual meaning of query words, to deliver more concise and accurate answers. The former, needs advanced Natural Language Processing (NLP) techniques while the latter requires semantic search.

Semantic search uses semantics, or the science of meaning in language, to produce highly relevant search results. In most cases, the goal is to deliver the information queried by a user rather than have a user sort through a list of

loosely related keyword results [31].

Question Answering systems which perform semantic search, are called semantic QA systems, and are powered by Semantic Web technology which was proposed by Tim Berners Lee in 2000. Kamath et al. (2013) declare that “the Semantic Web is an extension of the current web that allows the meaning of information to be precisely described so that Web content can be automatically processed by machines”. They emphasize that “all Semantic Web technologies are built on the strong foundation of XML (eXtensible Markup Language) standardized by W3C. Through XML, it has become possible to transfer data between systems as diverse as databases, Web services, semantic knowledge bases or end user applications in one common file format. Yet, XML by itself defines just the abstract syntax, the makeup of the information. It does not sufficient to convey meaning or the semantics of the structure of the data. The backbone of the Semantic Web is said to be the ontologies. Ontologies formally represent knowledge as a set of concepts within a domain, and the relationships among those concepts” [17].

In the following sections, more details about ontologies, Semantic Web as well as its architecture and technologies, and Linked Data that makes the Semantic Web a reality, are provided.

## 2.2 Ontology

Ontologies, which are formal specification of the terms in the domain and relations among them [15], play important roles in different fields of studies. In recent years, ontologies have become common on the World Wide Web (WWW). The ontologies on the Web, range from large taxonomies -ways of grouping things in hierarchical manner- categorizing websites (such as on Yahoo) to categorization of products to order and sell (such as on Amazon.com). WWW Consortium (W3C) is using ontology languages to encode knowledge on web pages in machine-understandable form to search for information. In many disciplines, domain experts develop standardized ontologies to share and annotate information in their fields [22].

SayedSayed and Muqrishi (2016) also emphasize on the importance of ontology in creation and management of knowledge in the Web. More clearly, they declare that “ontology provides shared knowledge which are rich in semantics and can be understandable for machines. Moreover, it is proposed as a solution for the problems that arise from using different terminologies to refer to same concept or using the same term to refer to different concepts” [28].

The rest of this section, defines the concept of ontology, describes its components, levels, and clarifies the relationship between the ontology and the Semantic Web.

### 2.2.1 Ontology definition

Navigli (2016) mentions that the concept of “ontology” is originated from the field of Philosophy. Ontology in this field, refers to the study of the nature of being, general existence or reality as well as fundamental classes and relationships of existing things. Gruber who was the first to introduced the term “ontology” to computer science, defines an ontology as “a formal and explicit specification of a shared conceptualization” [26]. Berneres-Lee (1999) clarifies this definition such that “ “shared” means the information described by ontology is commonly accepted by users; “explicit” requires the precision of both concepts and their relationships clearly defined; “conceptualization” is referred to an abstract model of a phenomenon” [4].

### 2.2.2 Degree of formalization in an Ontology

In the applications that use ontologies, different degrees of formalization in an ontology, are considered. Navigli (2016) introduces six levels for degree of formalization in ontologies changing from least to the most formalized knowledge resource:

- Unstructured text: just a text string with no structure.
- Terminology: a set of terms expressing concepts for a domain of interest. (e.g. “racing driver”, “sports team”.)
- Glossary: a terminology with textual definition for each term. (e.g. A “racing driver” is an athlete who participate in motor sport races.)
- Thesaurus: provides information about relationship between words like synonyms and antonyms. (e.g. “first driver” is synonym of “winner”.)
- Taxonomy: a hierarchical classification of concepts. (e.g. “Formula One racer” is a racing driver and “racing driver” is an athlete.)
- Ontology: A fully structured knowledge model, including things, their properties, and their relationship to other [26].

### 2.2.3 Ontology components

An ontology is composed of the following building blocks:

- Concepts (also called classes or types); are the core components of most ontologies. They represent meaningful groups of individuals that share common characteristics. (e.g. “Person” is a class of all people.)
- Instances (also known as individuals or objects); are the basic ground level of ontologies. They may model concrete objects like people, machines as well as abstract objects such as articles, occupation, etc.

- Relations; describe the way instances or individuals relate to each other. Three kinds of common relations are used in ontologies:
  1. The “*is-a*” relation (also called subclass-of or type-of) defines which objects are classified by which class. “Taxonomy” is a kind of ontology whose relations between concepts are all this kind. (e.g. A “sports event” is a subclass of a “social event”.)
  2. The “*instance-of*” relation which connects each instance to the concepts that represents its abstract counterpart. (e.g. “Germany” is an instance of “Country”.)
  3. The “*has-a*” or “*has-part*” relation (also called meteorology relation) that represents how objects combine to form composite objects. (A “racer driver” has a team.)
- Attributes (or properties); represent relations intrinsic to specific concepts (e.g. “name” and “birth date” of a person.)
- Restrictions on relations; are formally stated descriptions of what must be true for some assertion to be accepted as input. (e.g. The “*has-parent*” relation can connect only instances of the human concepts.)
- Rules and axioms: declarations in logical form that encode the overall theory that the domain ontology describes.
- Events which describe the changing of attributes or relations [23].

#### 2.2.4 Levels of an ontology

Navigli (2016) shows that depending on the degree of formalism, ontologies are classified to four sections:

1. “Top-level ontology” (also known as “upper ontology”); consists of general concepts and relations that are shared in all domains of interest. According to Navigli, “upper ontologies support semantic interoperability among many specific-domain ontologies by providing the most general concepts structured in hierarchy and optionally associating general rules and axioms about those concepts”. SUMO is among the several upper ontologies that have been proposed. It includes more than 1000 concepts and about 4000 relations between them.
2. “Middle or general-purpose ontology”; allows more specific concepts usually encoded in a domain ontology, connect to each other. It is designed to not only meet the needs of specific community but also provides terminological structure that can share between different communities. DBpedia knowledge base represents a kind of middle ontology. The DBpedia Ontology is a shallow, cross-domain ontology, which has been manually created based on the most commonly used info boxes within Wikipedia. The ontology currently covers 685 classes which form a subsumption hierarchy and are described by 2,795 different properties.



3. “Domain ontology” is a collection of vocabularies about concepts and their relationship in a domain. For instance, an ontology about the domain of “computer software” would model the “software developer” meaning. Unified Medical Language System (UMLS); is an example of domain specific ontology which includes a semantic network providing a categorization of medical concepts [21].
4. An “application ontology” which is developed for specific use or application that cannot be shared or used by another community. Application ontologies depend both on domains and on a specific task of interest and are typically used when crossing domains [26], such as “The Experimental Factor Ontology (EFO)” which is an application focused ontology, modelling the experimental variables in multiple resources at the EBI and open targets. This ontology has been developed to increase the richness of the annotations that are currently made in resources and to promote consistent annotation, to facilitate automatic annotation and to integrate external data [9].

### 2.2.5 Creation of an Ontology

Navigli (2016) mentions that there are two ways to create an ontology: manually or automatically. The former, refers to “ontology building” and the latter refers to “ontology learning”. Building of an ontology usually involves four iterating steps: analysis of required information; design of the concepts and the relations; implementation via a specific language, e.g. RDFS or OWL and finally testing the consistency of the designed ontology.

In contrast, ontology learning does not need to construct an ontology from the scratch, thus leading to reduce the cost of construction and maintenance, which often must be performed for a long period. The required steps for learning an ontology include “term extraction” to acquire domain terms; “taxonomy learning” in which concepts are hierarchically constructed; “relation learning” where non-taxonomic relations are learned; “learning of facts and axioms” which is the final step of learning.

The additional process which must be performed for both two methods is called “maintenance” which include keeping the updates, versioning and avoiding incompatibility with other ontologies [26].

Up to now, different approaches are used to build ontologies from scratch and obviously, there is not a general method for building any kind of ontology. Martin Dzbor et al. (2005) mention that there are some well recognized methodological approaches (e.g., METHONTOLOGY, On-To-Knowledge, and DILIGENT) that provide guidelines to help researchers to develop ontologies [8].

Among these, we introduce two efficient methodologies which are used as the main references for creating our ontology. The first one is “METHONTOLOGY” which has been considered as a well-structured methodology targeted for ontology engineers and researchers, introduced by Gomez-Perez, Fernandez and

De Vicente in 1996. This methodology is based on the experience acquired in developing an ontology in the domain of chemicals. It also highly recommends the reuse of existing ontologies.

“METHONTOLOGY” consists three kinds of activities (Management, support, and development). The order and the depth in which the activities should be done is provided with life cycle of the ontology which is shown in Figure 2.1.

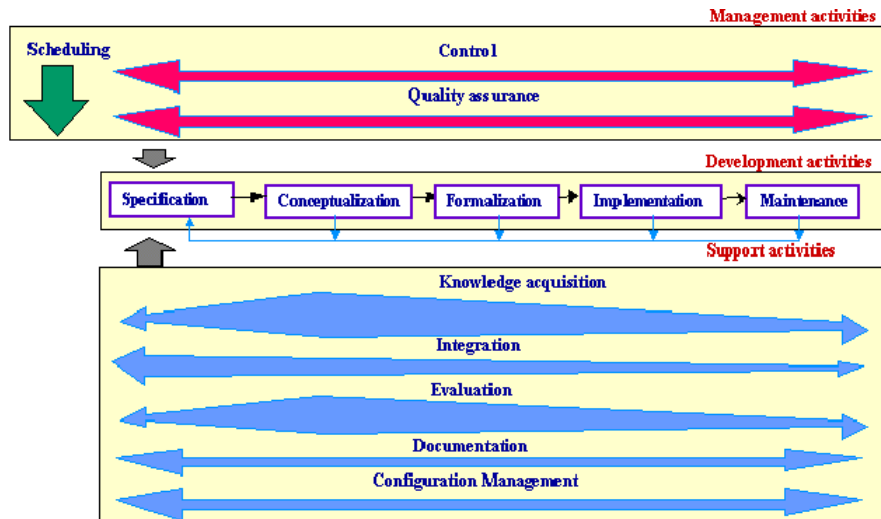


Figure 2.1: METHONTOLOGY life cycle [10]

As shown in this figure, scheduling is the first activity that is proposed to be performed while building a prototype. Then, this cycle moves forward sequentially through the development activities. Supporting activities as well as other management activities (control and quality assurance) are performed during the whole life cycle of the ontology. Figure 2.1 also shows that supporting activities like the knowledge acquisition, integration and evaluation are greater during the ontology conceptualization, and they decrease during formalization and implementation [10]. The following provides a brief description for each activity regardless of their order:

- Scheduling: to plan the main tasks, arrangement, and specifying required resources (people, software and hardware).
- Specification: to write answers to the competency questions or providing intermediate representations for describing the requirements that ontology should fulfill.
- Conceptualization: to build a conceptual model describing the problem and its solution.

- Knowledge acquisition: to capture required knowledge of the domain that ontology is designed for.
- Reusing existing ontologies (integration): to speed up the construction of the ontology.
- Implementation: to implement the ontology in a formal language to make it computable.
- Evaluation: to verify and validate the ontology before making it available to others.
- Documentation: to document all the steps to easily reuse or modify the ontology in future.
- Maintenance: to keep updates and avoid from incompatibilities with other ontologies.

“METHONTOLOGY” proposes “evolving prototype” life cycle which better fits the ontology life cycle. This prototype lets oncologists to add, remove or modify the definitions any time of the ontology life cycle [10].

The second methodology is called “NeOn” which is aimed at building of ontology networks by covering limitations of methodologies mentioned above and benefiting from their advantages. It practically, provides more detailed and precise guidelines for performing each activity of ontology building and facilitates building ontologies for software developers and ontology practitioners. NeOn methodology places important emphasis on reusing and re-engineering of both non-ontological resources (e.g. Glossaries, Dictionaries, Lexicons) and ontological resources (e.g. existing ontologies, ontology modules, design patterns) [8].

### 2.2.6 Ontology applications

There are several applications, in need of structured-knowledge, that benefit from ontologies. Question Answering systems and the Semantic Web are among these applications that take advantage of ontologies.

Ontology based QA systems accept NL queries and a given ontology as input, and return answers drawn from one or more Knowledge bases that subscribe to the ontology. Therefore, they do not require the user to learn the vocabulary or the structure of the ontology. They vary in two main aspects: (a) the degree of domain customization they require, which correlates with their retrieval performance, and (b) the subset of NL they are able to understand (full grammar-based NL, controlled or guided NL, pattern based) [7].

Figure 2.2 lists several examples of ontological QA systems as well as their used techniques in processing the questions.

Example of QA system	Used techniques
Aqua Log	Allows the user to choose an ontology and then ask NL queries with respect to the universe of discourse covered by the ontology
Power Aqua	QAS focusing on querying multiple semantic Web resources
QACID	Relies on an ontology, a collection of user queries, and an entailment engine that associates new queries to a cluster of existing queries
ORAKEL	Translates factual wh-queries into Flogic or SPARQL and evaluates them with respect to a given KB
GINSENG	Controls user's input via a fixed vocabulary and predefined sentence structures through menu-based options
PANTO	It takes an NL question as input and executes a corresponding SPARQL query on a given ontology
SPARQL2NL	In the side of converting a SPAQL query into natural language.
SWIP	In the side of converting a SPAQL query into natural language.
Pythia	Using ontology in the process of interpretation of user query
SQUALL	Using a controlled natural language for translation to SPAQL query
TBSL, LODQA	The user question is transformed to a template query. From the NL query to generate the SPAQL query using the template model.
Deep QA IBM Watson's system	Using unstructured and structured data (RDF format) to extract and score evidence

Figure 2.2: Ontological Question Answering Systems [7]

Ontologies have also become one of the main components of the Semantic Web- the web of data in which data can be understandable to machines. Webber (1997) discusses that the Semantic Web is based on the idea, common to the fields of Information Systems and Knowledge Management, that knowledge can be represented by a mapping of entities (things), their properties and their relationship to each other. It uses the tools of Information Systems and Knowledge Management, namely, “ontologies” and local specifications of entities and their properties, to achieve this [39]. Navigli (2016) outlines, “ontologies are considered as backbone of the Semantic Web. In fact, ontologies model knowledge to semantically annotate web pages, perform semantic search and create software agents that can understand user need” [26].

The next section, provides an overview of the Semantic Web and its relevant concepts.

## 2.3 Semantic Web

As mentioned briefly in the previous section, keyword based search techniques cannot answer correctly to the questions when there are various meanings of

identical concepts. Therefore, semantic search is employed to solve this problem by understanding the intention of user and the meaning of the concepts in the search query. The use of semantic technologies has improved search performance and accuracy by taking in to account the intent and conceptual meaning of the terms in query and data space where the answers originate from. The Semantic Web underpin these semantic technologies [38].

Web of data or Semantic Web, known as WEB 3.0, is a collaborative effort led by World Wide Web Consortium (W3C) with participation from many researchers and industrial partners. The goal of these efforts is to make web more understandable for machines. In the current Web, machines cannot derive meaning from the web contents (HTML documents), however in the Semantic Web, according to Bizer's definition in 2009, "information are given well-defined meaning, thus enabling machines and people work better in cooperation" [5].

### 2.3.1 Definitions of the Semantic Web

The term semantic implies "meaning" or "relating to the meaning or interpretation in language or logic". The Semantic Web is commonly described as the process for giving meaning to the web, or making the web understandable to machines [38].

Berners-Lee in statement of his dreams indicates that the meaning of data on the web can be discovered not only by people but also with computers. He believes that the Semantic Web, in which the web content is meaningful to computers, can assist the evolution of human knowledge [4].

The Semantic Web as a concept is defined from different perspectives; W3C (2001), has a machine-readable data view. They consider the Semantic Web as a vision: "the idea of having data on the web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications" [37].

Passian (2004) mentions different views regarding the Semantic Web. For example, Anutariya (2004) believes that the Semantic Web improves web search capabilities when it will be possible to access web resources by content rather than just by keywords.

Another view is defined by Euzenat (2004) who focuses on better annotation aspect. He states that "the idea of semantic web, supplies the informal web as we know it with annotations expressed in machine-process able form and linked together."

Aiding intelligent agents to retrieve and manipulate pertinent information, forming distributed databases, serving human in knowledge discovery from the Web; are among the other several views that arise from the Semantic Web concept [24].

### 2.3.2 Architecture of the Semantic Web

Tim Berners-Lee -known as inventor of World Wide Web and director of W3C clearly illustrates the architecture of Semantic Web in the form of Semantic

Web stack. This stack visualizes the hierarchy of languages, where each layer exploits and uses capabilities of the layers below. It shows how technologies that are standardized for Semantic Web are organized to make the Semantic Web possible. It also shows how Semantic Web is an extension (not replacement) of classical hypertext web [32]. Figure 2.3 illustrates the components of the Semantic Web stack.

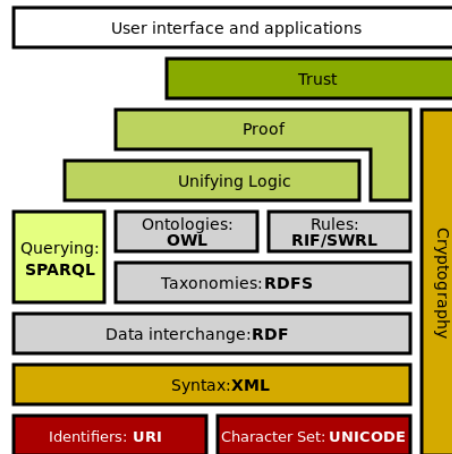


Figure 2.3: Semantic Web Stack (Tim-Berners-Lee, 2015)

Semantic Web stack consists of three main layers. Low layer includes hypertext web technologies which are Unique Resource Identifier (URI) and Character Set (Unicode). URIs are used to distinguish physical or abstract resources (things) from each other and Unicode is served to represent and manipulate text in many languages.

Immediately on top of that, XML (Extended Marked-up Language) is represented as a language which encodes documents in a structured format and readable for machines.

Middle layer consists of Semantic Web technologies; Both RDF (Resource Description Framework) and RDFS (RDF Schema) are formed based on XML syntax. RDF adds semantics to data which is structured by XML. RDF is a standard model of data interchange on the web which creates statements about resources in a form of triples (Subject, Predicate, Object).

RDFS is the basic schema language, which provides terminological knowledge for RDF in the form of classes and property hierarchies and semantic interdependencies. OWL (Web Ontology Language) at higher level of the stack, uses the RDFS syntax to represent more complex knowledge even the one which is only implicit in the domain of interest. OWL is a fully structured knowledge model including concepts, relations of various kinds and possibly rules and axioms [26].

SPARQL is a Semantic Web standard for querying RDF-based information

and for representing the results.

Alongside OWL, we have RIF (Rule Interchange Format) aimed at developing web standard for exchanging rules among disparate systems especially on the semantic Web applications. It allows describing relations that cannot be directly described using description logic used in OWL.

The top layers of stack (Logic, Proof, and Trust), deal with the logical and semantic validation of ontologies that are still ideas and should be implemented to realize the Semantic Web.

Moreover, cryptography layer covers most layers from bottom to top of the stack which ensures and verifies the Semantic Web statements come from trusted sources. “User Interface” and “Applications” constitute the final layer that enables human to use Semantic Web applications [32].

In the next section, the important Semantic Web technologies like RDF, RDFS and OWL are discussed.

### 2.3.3 The Semantic Web technologies

Rudolph et al. (2009) declare that “the Semantic Web has been conceived as an extension of the World Wide Web that allows computers to intelligently search, combine and process Web content based on the meaning that this content has to humans. In the absence of human-level artificial intelligence, this can only be accomplished if the intended meaning of Web resource is explicitly specified that is process able by computers”.

For this reason, they believe that “it is not enough to store data in a machine-process able syntax-like every HTMP page- but it is also required that this data is provided with a formal “semantics” that clearly specifies which conclusions should be drawn from the collected information. Semantic technologies have been developed to address this requirement”. Semantic technologies such as RDF, SPARQL, OWL, etc. enable people to create data stores on the Web, build vocabularies, and write rules for handling data [25].

#### RDF

The Resource Description Framework (RDF) is a formal language for modelling and representing information resources as structured data. The goal of RDF is to implement the vision of the Semantic Web in which web resources annotated with semantics, are easily understood by machines [26].

#### Basics of RDF

An RDF document, describes a directed graph, i.e. a set of nodes (subject or object) that are linked by directed edges (predicates). Both nodes and edges are labeled with identifiers (URIs) to distinguish them [25]. Figure 2.4 shows a simple example of a graph of two nodes and one edge. It represents that the

“first Race” of “Lewis Hamilton” was “Belgian Grand Prix” which occurred in year 1991.

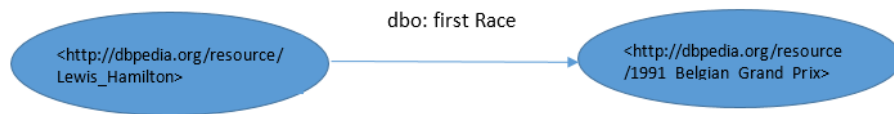


Figure 2.4: A simple RDF graph describing the relationship between the race driver and the GP race

dbo: is an abbreviation for “<http://dbpedia.org/ontology/>” which means “first Race” is a label defined at this http address.

Unlike XML documents, RDF information is not encoded in tree structure but in graph structure. Rudolph et al. (2009) introduce three reasons behind that; first, a graph consists of resources related to other resources, with no single resource having any intrinsic importance over another; while an XML document with a tree structure, typically contains nodes of information each with a parent node and at the root of the document is the highest-level node, which has no parent.

For example, the relationship between the driver “Lewis Hamilton” and the “Belgian Grand Prix” is a kind of information that does not in any obvious sense belong hierarchically below either of the resources. Another reason they mention for graph structure of RDF, is the fact that “RDF was intended to serve as a description language for data on the WWW and other electronic networks and in these environments, information is typically stored and managed in decentralized ways and it is very easy to combine RDF data from multiple sources. Moreover, related information items in trees might be separated by the strict structure: even if two XML files refer to the same resources, related information is likely to be found in very different locations in each tree. Therefore, Graphs in RDF are better suited for composition of distributed information sources” [25].

### Names in RDF: URIs

RDF uses so-called “Uniform Resource Identifiers” (URIs) as names to clearly distinguish resources from each other. URIs are a generalization of URLs (Uniform Resource Locators), i.e. of Web addresses as they are used for accessing online documents. URLs are valid URIs and indeed can be used as identifiers in RDF documents that talk about Web resources, however in numerous other applications the goal is to exchange information about different kinds of objects not about web pages.



In general, this might be any object like books, places, people, events, relationship among such things or all kinds of abstract concepts that has a clear identity in the context of a given application. Such resources can obviously not be retrieved online and hence their URIs are used exclusively for unique identification [25].

As shown in Figure 2.4, nodes, and edges in RDF graphs both are labeled with URIs to distinguish them from other resources.

### Data values in RDF: Literals

Data values in RDF are represented by so-called “literals”. These are reserved names for RDF resources of a certain datatype. A sequence of characters generally describes the value of every literal. The interpretation of such sequence is then determined by a given “datatype”.

Knowing the datatype is crucial for understanding the intended meaning: the character sequences “42” and “042”, refer to the same natural number but to different text strings. On the other hand, there are literals with no datatype. Such “untyped” literals are always interpreted as text strings. An example of untyped literal is shown in Figure 2.5 where the driver’s name is described as a simple text string [25].



Figure 2.5: A simple RDF graph with literal for describing data value

“dbp” is an abbreviation for this address: “http://dbpedia.org/property/”

As can be seen in Figure 2.5, when drawing RDF graphs, rectangular boxes are used to distinguish literals from URIs which are shown by oval shapes. Another special trait of literals is that they may never be the origin of edges in an RDF graph. In practice, it means that we cannot make direct statements about literals. Moreover, it is not allowed to use literals as labels for edges in RDF graphs, since it is hard to see what could be intended with such a labeling.

### Syntax for RDF

Rudolph et al. (2009) mention that there must be specific syntax formats for RDF graphs. They reason that “the way of representing RDF in diagrams is easy to read and still precise, yet it is not clearly suitable for processing RDF in computer systems. Practically relevant data sets with thousands or millions of nodes obviously cannot be stored and communicate in pictures.” They declare that “there are different ways of representing RDF by means of

character strings that can easily be kept in electronic documents. This requires splitting the original graph into smaller parts that can be stored one by one. Such a transformation of complex data structures into linear string is called “serialization” “[25].

### Triple serialization

In this way of representation, each edge corresponds to an “RDF triple”. There are three distinguished parts in each edge; they are called “subject”, “predicate” and “object” respectively. “Subject” node denotes the resource being described and is represented by a URI. The edge (also called “predicate”) denotes a property of the subject or a relation between the subject, and the object. The predicate is generally a term from a well-known vocabulary or ontology represented by a URI. The node “object” denotes the value of a property or another resource which is the target of the relation.

For instance, from Figure 2.4 and Figure 2.5 we distinguish these three parts as following shown in Table 2.1.

Subject	Predicate	Object
dbr:Lewis Hamilton	dbo: first Race	dbr:1991 Belgian Grand Prix
dbr:Lewis Hamilton	dbp: name	“Lewis Hamilton”

Table 2.1: Sample triple statements in RDF

Turtle, N3 and N-triples are triple syntax formats that were developed for RDF. N-triples is a less complicated part of N3 as possible syntax for RDF which is very simple, easy to parse, has a line-based format and is not as compact as Turtle.

An example of N-triples format of RDF document including three sentences is shown in Figure 2.6. In this example, URIs are abbreviated using prefixes of the form “prefix” and are no longer enclosed in angular brackets.

```
@prefix dbr: <http://dbpedia.org/resource/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix dct: <http://purl.org/dc/terms/>.
@prefix dbo: <http://dbpedia.org/property/>.

dbr: Michael Schumacher    foaf:name    "Michael Schumacher "@en.
dbr: Michael Schumacher    dbo: first Win  "1992"^^<http://www.w3.org/2001/XMLSchema#integer>.
dbr: Michael Schumacher    dct: subject   dbr: Category: German_Formula_One_drivers.
```

Figure 2.6: A sample RDF document with N-Triples format

In addition to Triple representation which is more suitable for human use,

in practice, there is a main syntax for RDF known as “RDF/XML” which Many programming languages offer standard libraries for processing it.

This syntax is XML-based serialization and offers several additional features and abbreviations that are convenient to represent advanced features. Figure 2.7 describes the triples from Figure 2.4 and Figure 2.5 in RDF/XML format.

```
<? xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dbo="http://dbpedia.org/ontology/"
  xmlns:dbp="http://dbpedia.org/property/" >
  <rdf:Description rdf:about="http://dbpedia.org/ontology/Lweis_Hamilton">
    <dbp:name>Lewis Hamilton</dbp:name>
    <dbo:first Race rdf:resource="http://dbpedia.org/resource/1991_Belgian_Grand_Prix" />
  </rdf:Description>
</rdf:RDF>
```

Figure 2.7: A sample RDF/XML document

### 2.3.4 RDFS

It is possible to give identity and structure to data using URIs and RDF, but to add semantic to data, ontology is needed. As described before, an ontology, represents knowledge as a hierarchy of concepts in a domain using a shared vocabulary to specify classes (concepts), properties and relationships of those concepts.

RDF Schema provides basic constructs to define an ontology to specify RDF real data; it allows to define classes, properties, and their subsuming hierarchies along with the domain and the range of each property [20].

In RDFS vocabulary, there are elements such as `rdfs:class` which define “class” but instances of a class are defined using `rdf:type`. There are also elements that define properties via `rdf:Property` and property restrictions via `rdfs:domain` and `rdfs:range`. hierarchical relationship is denoted by sub classes and super classes via `rdfs:subClassOf` and `rdfs:superClassOf`. Figure 2.8 presents an example of RDF/XML document stating that “Race Driver” is a class and subclass of “Athlete” and “Lewis Hamilton” is an instance of this class and has a “name” property which can accept literal values and all the instances of class “Race driver” can have the “name” property.

### 2.3.5 OWL

OWL is the latest standard in ontology languages from the World Wide Web Consortium (W3C). It is built on RDF and RDFS and provides additional vocabularies for defining classes and relations.

```

<? xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:dbo="http://dbpedia.org/#">
<rdf:Description rdf:about="http://dbpedia.org/#Lewis_Hamilton">
<rdf:type rdf:resource="http://dbpedia.org/#RacingDriver"/></rdf:Description>
<rdf:Description rdf:about="http://dbpedia.org/#Racing_Driver">
<rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
<rdfs:subClassOf rdf:resource="http://dbpedia.org/#Athlete"/></rdf:Description>
<rdf:Description rdf:about="http://dbpedia.org/#Athlete">
<rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/></rdf:Description>
<rdf:Description rdf:about="http://dbpedia.org/#name">
<rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
<rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
<rdfs:domain rdf:resource="http://dbpedia.org/#Racing_Driver"/></rdf:Description></rdf:RDF>

```

Figure 2.8: An example of RDF/XML document using rdfs elements

RDFS suffers from many weaknesses, that leads to an extension to the ontology upper layer. For instance, RDFS cannot describe resources in sufficient details because there are no localized constraints for defining range and domain. It has also no reasoning support and no cardinality constraints, transitive, or symmetrical properties [28]. Therefore, OWL was created from the need to extend RDFS to increase its expressivity, thus adding a consistent number of constructs useful to better formalize a domain.

To allow usability by various users, OWL provides three increasingly expressive sub-languages: OWL-Lite, OWL-DL and OWL-Full [20].

“OWL Lite” which is contained in two other sub-languages, is decidable, less expressive and in worst case, it has computational complexity; “OWL DL” contains Lite version and similarly is decidable but, most software tools fully support it; “OWL Full” compared with those two sub-languages is very expressive and contains all RDFS, undecidable and hardly supported by any software tools. It is semantically difficult to understand and work with [25].

### 2.3.6 SPARQL

SPARQL is the query language of the Semantic Web. It stands for SPARQL Protocol and RDF Query Language. According to the definitions provided by

W3C; “SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs” [34].

## 2.4 Linked Data

Bizer (2008) outlines that “Semantic Web has been brought to being by the maturing of the Semantic Web technology stack and by publication of an increasing number of data sets according to the principles of Linked Data” [6].

There are different opinions on definition of Linked Data; Tim Berners-Lee, the person credited with coining the terms “Semantic Web”, believes that Linked Data is not a separated part and defines it as “the Semantic Web done right”, however a widely-held view is that the Semantic Web is made up of Linked Data implying that Linked Data is the parts of the Semantic Web.

Regardless of different interpretations, technically “Linked Data” refers to data published on the Web in such a way that has explicit meaning, machines can understand, read, and manage it, external datasets can be linked to it and it can also be linked to other external datasets [5].

In the following section, a short review of the purpose and principles of Linked Data as a fundamental part of the Semantic Web is provided.

### 2.4.1 Why Linked Data

Bizer et al. (2009) believe that traditionally, most structure and semantics of data is sacrificed, since the published data on the web is mostly available in formats such as CSV, XML, or HTML tables. In the conventional hypertext, Web, in which users or machines can move from one document to another by using Hypertext links (typed links), the relationship among documents has remained implicit, since the concepts (entities) in HTML documents cannot be connected to the related concepts by typed links.

However, they represent that “in recent years the global information space has been extended to both linked documents and linked data. As the result of this evolution, a set of best practices for publishing and connecting structured data on the Web known as “Linked Data” have been emerged. In this space, data from diverse domains can be connected and queried; domains such as people, companies, books, scientific publications, films, music, television and radio programs, genes, proteins, drugs and clinical trials, online communities, statistical and scientific data, and reviews” [5].

Linked Data also empowers search engines. Search engines by working on the Linked Data, can provide sophisticated query capabilities, like “those provided by conventional relational databases, because the query results themselves are structured data, not just links to HTML pages, they can be immediately

processed, thus enabling a new class of applications based on the Web of Data” [6].

### 2.4.2 The principles of Linked Data

Berners-Lee (1999) outlines a set of “rules” for publishing data on the Web in a way that all published data becomes part of a single global data space:

1. Use URIs as names to clearly distinguish things from each other.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards like RDF.
4. Include links to other URIs, so that they can discover more things.

While HTML provides a means to structure of documents and link documents on the Web, Resource Description Framework (RDF) provides a structure for the data on the web and link data with a graph-based data model. Linked Data employs these standards to publish structured data on the Web and to connect data between different data sources, effectively allowing data in one data source to be linked to data in another data source [5].

### 2.4.3 Linking Open Data project

The most visible example of adoption and application of the Linked Data principles has been the Linking Open Data project (cloud). The original and ongoing aim of the project is to bootstrap the Web of Data by identifying existing data sets that are available under open licenses, republish these in RDF on the Web according to the Linked Data principles, and interlink them with each other [5].

Figure 2.9 shows an indication of the range and scale of the Linking Open Data “cloud”, this figure shows main interlinking hubs are data sources such as DBpedia and Geonames. DBpedia extracts RDF triples from the “Info-boxes” commonly placed along side of Wikipedia articles, and makes these available on the Web in RDF to be crawled or queried with SPARQL, whereas Geonames provides RDF descriptions of millions of geographical locations worldwide. As these two data sets provide URIs and RDF descriptions for referring to many entities, many other data sets are using them as the main references, therefore they have been developed into hubs where an increasing number of other data sets are connected [5].

### 2.4.4 Linked Data applications

In addition to publishing and interlinking data sets, there are many applications that exploit Linked Data.

Linked Data browsers, which allow users to navigate between data sources by following links expressed as RDF triples; Linked data search engines like

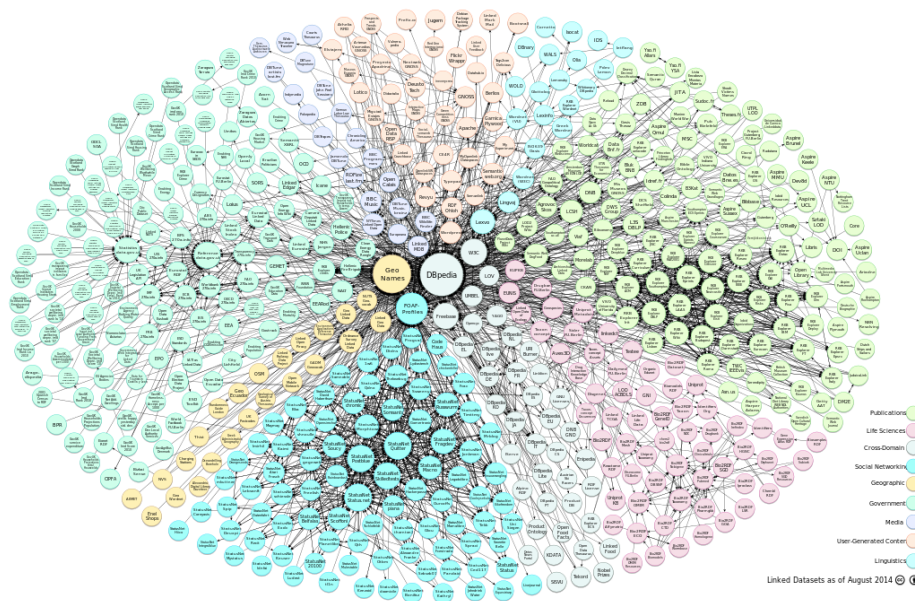


Figure 2.9: Linking Open Data cloud diagram [29]

Falcons and SWSE that provide keyword-based search services and similar to Yahoo and Google are oriented towards human; lookup indexes like Swoogle which serves the needs of specific applications; domain specific applications offer more domain-specific functionality by “mashing up” data from various Linked Data sources. All These services enable humans and machines to locate and query Linked Data that has been published on the Web [6].

## 2.5 Knowledge-base QA systems

Referring to the “ontology” Section, like ontologies that can be targeted at either all domains or a specific domain; QA systems can also be divided in two types based on the domains of questions:

1. Open-domain QA systems: deals with the questions which are related to every domain. These systems, mainly have more data available from which the system extract the answer. It can answer any question related to any domain but with very low accuracy as the domain is not specific [19].
2. Closed-domain QA systems: refers to specific domain related questions and can be seen as an easier task because NLP systems can provide domain-specific knowledge. It has very high accuracy but limited to single

domain. The example of such system is medicines or automotive maintenance [19].

QA systems also use two major paradigms of question answering to provide answers to the questions in different formats; known as IR based question answering and knowledge-based question answering.

IR-based (Information Retrieval-based) question answering mostly rely on a large amount of unstructured information (like documents, web pages, snippets, etc.) available on the Web or in specialized collections. They generally answer the questions in three phases:

- “Question processing” where the answer type (often a named entity e.g. person or a place) is determined and the query is formulated to send to the search engine.
- “Passage retrieval” in which the relevant passages (that contain query keywords) are obtained and ranked.
- “Answer processing” where candidate answers from the passages are extracted and ranked.

Many open-domain QA systems use this paradigm where the degree of similarity between query and documents is assessed and the candidate answers are extracted from the most relevant passages. (with highest score of similarity)

In contrast, closed-domain QA systems mostly use the knowledge base paradigm, since they translate user’s question into a database query, then this query is applied to the database for providing the answer [2].

The QA systems which make use of knowledge base paradigm, take advantage of more structured forms of data on the web, like RDF, XML and relational databases. They answer a natural language question by mapping it over a structured database. “BASEBALL” (Green et al., 1961) was one of the earliest question answering systems which developed to answer user questions about Baseball stats and games from a structured database of Baseball [14].

Knowledge-based QA systems build a semantic representation of the query (for example using a query language like SQL or SPARQL) to access the required information in databases of facts. Depending on the scale and complexity of the application, these databases can be full relational database and quite complex (e.g. full of scientific or geo-spatial data) which need powerful logical queries or they can be databases including simple relations, known as triple stores, such as DBpedia and Freebase which are popular ontologies and build structured data from Wikipedia articles in format of RDF triples [16].

Like a relational database, one stores information in a “triple store” and retrieves it via a query language, but unlike a relational database, a triple store is optimized for the storage and retrieval of triples. In addition to queries, triples can usually be imported or exported using Resource Description Framework (RDF) and other formats.



There is also a more generalized structure than a triple store, known as “Graph database” which uses graph structures with nodes, edges, and properties to represent and store data. Graph databases might provide index-free adjacency, meaning every element contains a direct pointer to its adjacent elements, and no index look-ups are necessary [35].

Using a graph database as the source of answers in a question answering system, have several advantages over using a relational database [13]:

1. A graph database generally uses graph structures for semantic queries to represent and store data. A key concept of the system is the graph, which directly relates data items in the store. The relationships allow data in the store to be linked together directly, and in many cases retrieved with one operation, whereas in relational databases, links between data are stored in the data, and queries search for this data within the store and use the “join” concept to collect the related data which is costly for complicated queries that need complex “join” operations.
2. Graph databases, by design, allow simple and fast retrieval of complex hierarchical structures that are difficult to model in relational systems.
3. SPARQL is one of the main query languages that is used for querying RDF data in graph databases and is introduced as one of the components of the Semantic Web Stack, while for relational database, “SQL” is the main query language which is not suitable for querying RDF data.

## Chapter 3

# Development of the QA system

In this chapter, we describe the development process of QA system, with focus on “Ontology Construction” and “Answer Extraction” modules mentioned in the proposed model in Chapter 1; first, we address the offline stage, where design of an ontology and its implementation with the help of semantic technologies and a knowledgeable, are described in Sections 3.1 and 3.2. Then, in Section 3.3, the online stage is addressed where we explain how the “Answer Extraction” module works and interact with other parts. Lastly, implementation of a working prototype for the QA system is explained in Section 3.4.

### 3.1 Design of the ontology

In the development process of the QA system which makes use of semantic technologies, ontology plays important role from two aspects: first, ontology is used in the process of interpretation of user question and translation to a formal query; second, it is used as a structured knowledge model to create a semantic knowledge-base. This KB can be used as a machine-readable resource for the system to retrieve the answers.

Regarding the mentioned roles for the ontology in this study and referring to concepts of ontology and QA systems described in Chapter 2, we consider the following specifications for the target ontology:

- It should have the most degree of formalization, including things, properties, and their relationship to other.
- It should be a domain-ontology, as we want to represent knowledge in a specific domain (Formula One sport).
- As the QA system is targeted at answering the questions in a specific domain, it is considered a closed-domain system and it is preferred to use

knowledge-based paradigm for answering the questions. For this purpose, the ontology can be implemented through a knowledge base of facts to provide more concrete and precise answers for the system.

In the following, the main steps for design of target ontology are described. In the previous chapter, we introduced two ways to develop an ontology: “building” versus “learning” and two methodologies to develop an ontology: “METHONTOLOGY” and “NeOn”. For this project, we build an ontology from scratch and for the process of design and implementation of the ontology, we mainly follow the steps outlined in the mentioned methodologies.

In the design process, generally METHONTOLOGY approach is used which introduces the required activities in each step for building an ontology, but for performing important activities, like specification of requirements and reusing existing ontologies which METHONTOLOGY does not specify required and detailed guidelines to perform the tasks; we follow the NeOn guidelines to make use of its advantages over the other one.

Moreover, similar to the life cycle of METHONTOLOGY, for this project the “evolving prototype” is chosen as the appropriate life cycle for building the ontology, in which we can go back from any state to other if some definition is missed or wrong.

The following sections describe the development process of Formula One ontology based on these methodologies.

### 3.1.1 Specification

This activity is a collection of requirements that the ontology should fulfill. The result of this activity is the Ontology Requirements Specification Document (ORSD). The following tasks are required to do this activity:

1. To identify the purpose, level of formality and the scope of ontology.
2. To identify the intended users and intended uses.
3. To identify and validate the requirements which the ontology should satisfy after being formally implemented.
4. Extraction of required terminology for building ontology.

To do Task 1, the potential need for developing a knowledge model in domain of Formula One sport was specified by consultancy with the supervisor of the project.

To determine degree of formality in the ontology, we referred to the definitions provided by METHONTOLOGY approach which relate it to the formality that will be used to codify the terms and their meaning. Uschold and Gruninger (1996) classify the level of formality in a range of “informal”, “semi-formal” and “rigorously formal”, depending on whether terms and their meaning are codified in a language between natural language and a rigorous formal language [36].

Since the implementation language of the target ontology will be RDFS, the level of formality will be rigorously formal. To identify the scope of ontology, we used the questions about domain and scope of the ontology, based on the “Ontology Development 101” guide. This guide addresses the scope of ontology by providing answers to the basic questions such as: “*What is the domain that the ontology will cover?*”, “*For what are we going to use the ontology?*”, “*Who will use and maintain the ontology?*”, as well as identifying the ontology competency questions [22].

To do Task 2, through consulting with the supervisor, we specified the target group and intended uses of the ontology. The results are documented in the ontology requirements specification document.

To do Task 3, most of the methodologies like “METHONTOLOGY” and “On-To-Knowledge”, propose the use of competency questions as a useful technique to gather the ontology requirements. “Competency questions” are defined as natural language questions that the ontology to be built should be able to answer.

In this study, it is assumed that the potential users ask for statistical data about Formula one races, drivers, teams, and seasons. Other kind of questions like predictive or descriptive ones are not the target questions of our QA system. In addition, the questions are asked in natural language format in “English”. Considering these assumptions, we prepared 40 questions for the specification phase and tried to make them various by addressing different entities or concepts in the questions in terms of type and number.

Moreover, we prepared questions in a way that range from easy to difficult. Answer(s) to easy questions can be a simple fact or multiple facts directly available in Wikipedia tables; while answer(s) to difficult questions often need statistical calculations on available information in Wiki- tables. For example, the question *Who won the Spanish GP in 2013?* is considered “easy” that can be answered by simple facts directly available in Wiki- table, but the question *Which team has the most wins since 2010?* is considered as a difficult question requiring calculations on the data.

Regarding the points mentioned above for adding variety to the questions, we classified them in three categories: factoid, list, and aggregation questions. Factoid questions are the ones that can be answered with simple facts expressed in short text answers, similarly, list questions can be answered with a list of simple facts and aggregation questions are more complex questions that can be answered either with a single fact or a list of facts. A sample list of competency questions, as well as their answers and categories are presented in Figure 3.1.

In the next step, we tried to validate the competency questions through consulting with the supervisor and used his recommendation and suggestion to choose the most appropriate questions regarding the categorizations mentioned above. The correctness and consistency of the answers are checked by referring to the available resources on the Wikipedia info-boxes and tables.

To perform Task 4; extraction of required terminology, the main objects in the domain of Formula One were recognized from the terms used in competency questions and the answers: drivers, races (Grands prix), seasons and teams.

Question ID	Competency Question	Answer	Category
CQ1	Who won the Spanish Grand prix in 2013?	Fernando Alonso	Factoid
CQ2	Which team won Canadian Grand Prix in 2014?	Red Bull Racing	Factoid
CQ8	How many championships were recorded for Michael Schumacher?	7	Factoid
CQ12	In which Grand prix did Jenson Button win for the first time?	2006 Hungarian Grand Prix	Factoid
CQ21	List the teams that Kimi Raikkonen had contract with?	Sauber, McLaren, Ferrari	List
CQ22	List the races where Daniel Ricciardo was the fastest driver?	2015 Singapore, 2014 Abu Dhabi, 2015 Monaco, 2015 Hungarian GP	List
CQ26	Which drivers became champions in seasons 2009 and 2010?	Jenson Button, Sebastian Vettel	List
CQ30	Who has more fastest laps than Niki Lauda? (Name three drivers)	Sebastian Vettel, Lewis Hamilton, Kimi Raikonen	Aggregate
CQ31	Which team has won the most Grands Prix since 2010?	Red Bull Racing (43 wins)	Aggregate
CQ33	Which team has the most podiums in the history of F1?	Ferrari	Aggregate

Figure 3.1: A sample list of competency questions and answers

Then to extract pre-terminology for instances of these objects, we made use of the information provided in the Wikipedia info-boxes and tables which is related to the identified objects in domain of Formula One. This terminology will be formally represented in the form of concepts (entities), attributes and relations during conceptualization phase. Examples of the terms related to “Formula one driver” (racer) are shown in Table 3.1.

Related terms to Formula One racer	
personal Information	Formula One Career
name	races
birth Date	wins
birth Place	poles
nationality	podiums
occupation	championships
	fastest Laps
	first Win
	last Win
	first entry
	last entry
	points
	position
	season
	GrandPrix
	team

Table 3.1: Sample terms related to Formula One racer

At the final step of Task 4, using the information sources mentioned above, we outlined a number of instances of each recognized object and listed them in Table 3.2. These example objects will be represented as instances in the conceptual phase.

Objects			
Formula One Racer	Formula One Season	Grand Prix	Formula One Team
Lewis Hamilton	Season 2010	2010 Canadian GP	Red-Bull
Mark Webber	Season 2011	2011 Abu Dhabi GP	McLaren
Sebastian Vettel	Season 2012	2012 Australian GP	Ferrari
Jenson Button	Season 2013	2013 Japanese GP	Mercedes Benz
Nico Rosberg	Season 2014	2014 Monaco GP	Williams
Felipe Massa	Season 2015	2015 Brazilian GP	Renault
Jenson Button	Season 2016	2016 Hungarian GP	Sauber

Table 3.2: Examples of objects

After fulfilling all the tasks, the output of the Ontology Specification activity is the Ontology Requirements Specification Document which is presented in Figure 3.2.

Figure 3.2: Formula One Reference Ontology Requirement Specifications Document

<b>Formula One Reference Ontology Requirements Specification</b>			
<b>1.Purpose</b>			
The purpose of building the Reference Ontology is to provide a knowledge model of Formula One sport domain that could be used in an answering engine to provide answers to questions about Formula1.			
<b>2.Level of Formality</b>			
The ontology is implemented in RDFS language which is a formal ontology language			
<b>3.Scope</b>			
The ontology focuses on statistical information available in domain of Formula one sport. The level of granularity is directly related to the competency questions and terms identified.			
<b>4.Intended Uses</b>			
Search for Formula one information and Update for Formula One statistics in search applications like an answering engine.			
<b>5.Intended Users</b>			
5.1 People or organizations who are interested in Formula One news and statistics including Formula One fans, Athletes specially Formula One Racers and constructors.			
5.2. Search engine developers who use the ontology in their system to provide answers to questions related to Formula One.			
5.3 Developers of existing ontologies and knowledge bases to integrate and update F1 data.			
<b>6.Groups of Competency Questions</b>			
<b>CQ. Cat.1 Factoid questions (samples):</b>			
Who won the Spanish Grand prix in 2013?			
Which team won Canadian Grand Prix in 2014?			
What was course length of the Japanese Grand Prix in 2010?			
<b>CQ. Cat2. Listing questions (samples):</b>			
which team and driver became champion in season 2012?			
List the teams that Sebastian Vettel had contract with?			
List the races where Daniel Ricciardo was the fastest driver?			
<b>CQ. Cat3. Aggregating questions(samples):</b>			
Which season had the most races?			
How many times have Kimi Raikkonen become second driver with McLaren?			
Which countries have hosted the least GPs since 2010?			
<b>7.Pre-Glossary of Terms</b>			
<b>Sample Terms related to Formula One Racer</b>			
<b>Personal info:</b>		<b>Formula One Career:</b>	
Name		wins, poles, podiums	
Nationality		fastest Laps, championship	
Birth date		Grand Prix, F1 Season	
Birth place		first win, last win, first entry, last entry	
<b>Objects with sample instances</b>			
<b>Formula One Racer</b>	<b>Formula One Season</b>	<b>Formula One Grand Prix</b>	<b>Formula One Team(Constructor)</b>
Lewis Hamilton	Season 2010	2010Canadian GP	Red-Bull
Mark Webber	Season 2011	2011Abu Dhabi GP	McLaren
Sebastian Vettel	Season 2012	2012Australian GP	Ferrari
Jenson Button	Season 2013	2013Japanese GP	Mercedes Benz
Nico Rosberg	Season 2014	2014Monaco GP	Williams

### 3.1.2 Conceptualization

The goal of conceptualization activity is to structure the domain knowledge in a conceptual model that describes the problem and its solution in terms of the domain vocabulary which is human readable [10]. The following tasks are determined for the conceptualization step:

1. To provide complete Glossary of Terms (GT) in the intended domain.
2. To describe GT in a Data Dictionary including concepts, instances, attributes and relations.
3. To describe instance attributes and relations in tables of “Instance attribute” and “Instance relation” and to describe instances in tables of “Instances” [12].
4. To build a preliminary conceptual model based on the gathered knowledge in the previous tasks.
5. To build the extended conceptual model by improving the preliminary model.

The generated conceptual model allows the final users: (a) to figure out if an ontology is useful and usable for a given application without inspecting its source code; and (b) to compare the scope and completeness of several ontologies, their re-usability and share-ability by analyzing the knowledge expressed in each Intermediate representation (IR). IR is considered as the result of each task mentioned above [10].

In this project, all the first four tasks are carried out with the help of DBpedia ontology and Dbpeida knowledgebase in both Class and Instance levels. The result of final task will be the modified conceptual model in the domain of Formula One. The following, describe these tasks in more details:

According to “METHONTOLOGY”, starting point for conceptualization is providing complete Glossary of Terms (GT) including concepts, instances, relations and properties. Fernandez (97) states that: “the GT identifies and gathers all the useful and potentially usable domain knowledge and its meaning” [10]. In this study, many terms of the intended ontology have already been identified in the document of specification phase (ORSD). The extracted terms in the defined ORSD are mainly driven from Wikipedia articles. There are some strong reasons behind:

Wikipedia is the most popular online encyclopedia and ranks among the top ten visited sites [30]. Wikipedia articles consist of different types of structured data such as info-boxes, tables, lists and categorization data. This structured data is extracted by the extraction frame work of the DBpedia project and is turned into a rich knowledge base. The structure of the DBpedia knowledge base is maintained by the DBpedia user community. Most importantly, the community creates mappings from Wikipedia information representation structures to the DBpedia ontology [18]. The DBpedia Ontology is a multi-domain



ontology that serves as a structural backbone for this data. This ontology was manually created and has been maintained by the DBpedia team from the FU Berlin [30].

To speed up the conceptualization activity, we decided to follow the approach of reusing existing ontologies. “The NeOn Glossary of Activities” defines “ontology reuse” as the activity of using an ontology or an ontology module in the solution of different problems [8]; the problems like design of an ontology-based answering engine.

To follow this approach, reuse of DBpedia ontology was considered as an appropriate choice for several reasons; first it is a multi-domain ontology which provides knowledge in different domains. Domain-closed ontologies such as “Formula One ontology” which is intended to provide knowledge of a certain domain (sports domain) can reuse a part of DBpedia ontology which defines the relevant concepts (entities) and relations in Formula One domain.

Second, as mentioned above, DBpedia ontology is the one which its statements and definitions are the mappings from Wikipedia articles where the glossary of terms for Formula One ontology was identified. Third, it covers most of the competency questions which were determined in the specification phase.

Furthermore, accessing to this ontology is easy and free; it does not impose any time limitation or cost. It is also well documented and provides source of data freely in different formats of RDF (N3, XML, Json, N-Triples, Jason). As the target language in implementing of the Formula One ontology is RDFS, this kind of ready formats can save time in the implementation phase. The definitions and naming conventions of DBpedia ontology about Formula One is very similar to the ones that the intended ontology is going to use. Finally, DBpedia ontology is regarded as a reliable ontology where very well-known ontologies or projects have been reused it [8].

Next step in conceptualization, is building a “data dictionary” where for each identified concept in the domain, the ontology builder should fill in the following fields of the dictionary: concept name, synonyms, description which provides meaning of the concept, instances of the concept, Class hierarchical level, Instance attribute and Instance relations which provide relevant properties that describe the instance of concepts.

Figure 3.3 shows the data dictionary which was built based on DBpedia ontology in Formula One (F1) domain. The selected concepts (Classes) of DBpedia ontology are the ones which are directly related to Formula One category. In instance level, entities with the same identifier of DBpedia knowledge base are reused.

Concept Name	Synonyms	Description	Sample Instances	Class hierarchy level	Instance Attribute	Instance relations
Grand Prix	Formula 1 race, F1GP, F1 Grand Prix	Grand Prix is an Auto racing event, a tradition dating back to the first decade of the 20 <sup>th</sup> Century and the Grand Prix motor racing of the 1920s and 1930s. Formally for a race to be called Grand Prix, it should have a race distance of at least 300km(190mi).	2010Abu Dhabi Grand Prix 2012Australian Grand Prix 2008Japanese Grand Prix 2013Hungarian Grand Prix 2011Brazilian Grand Prix ...	Subclass of sports Events	Course, distance, distance Laps	country, fastestDriverTeam, FirstDriver, second Driver, third Driver, firstDriverTeam, pole Driver, Pole Driver Team, fast Driver, fast Team, first Team, second Team, Third Team, pole Team
Formula One Racer	F1 driver	Is Formula One racing driver	Lewis Hamilton Mark Webber Sebastian Vettel Jenson Button Nico Rosberg Felipe Massa	Subclass of Racing Driver	Name, Championships, podiums, wins, races, fastest laps, poles, last position,	first race, first win, last win, last race, current team
Formula One Team	F1 Constructor, Entrant organization	In F1 racing, the terms constructor, team, and entrant have become synonyms since 1981 season when FIA have required that Formula One entrants own the intellectual rights to the chassis that they enter. Every team in Formula One must run two cars in every session in a Grand Prix weekend, and every team may use up to four drivers in a season.	Red-Bull McLaren Ferrari Mercedes Benz Williams Renault Toro Rosso BMW Sauber ...	Subclass of Sports Team	Name label	Is first Team of, is pole Team of, is fastest Team of, is third Team of, is second Team of, is first Driver Team of, is fastestDriverTeam of

Figure 3.3: Data dictionary in the domain of Formula One (based on DBpedia ontology)

To perform Task 3, the tables of “instance attributes” and “instance relations” are provided. Some examples like “podiums” and “distance laps” as attributes and “(has) first driver” and “(has) pole team” as relations are shown in the following tables:

<b>Name</b>	podiums
<b>Description</b>	The Podium is the name given to the rostrum on which prizes are awarded to the three highest placed drivers or teams at the end of a Grand Prix. “Podiums” refer to the number of times the driver (or the team) has been among the three top drivers or teams.
<b>Value type</b>	natural
<b>Range of values</b>	Non-negative integers
<b>Domain</b>	Formula One Racer, Formula One Team

Table 3.3: Table of instance attribute: podiums

<b>Name</b>	distance Laps
<b>Description</b>	It is the distance travelled by a car over a lap. This distance may vary from lap to lap as the car takes different lines over different laps. However, in most cases, it is the length of the racing line along the circuit. In DBpedia ontology number of laps is used as unit of measure for this attribute.
<b>Value type</b>	real
<b>Unit of measure</b>	Laps, Km, Mile
<b>Range of values</b>	integers
<b>Domain</b>	Grand Prix

Table 3.4: Table of Instance attribute: distance Laps

<b>Name</b>	(has) first Driver
<b>Description</b>	The driver who stands on first position of the podium (winner of that specific GrandPrix) is called first driver. The relation “(has) first Driver” is a relation between F1 Racer and Grand Prix.
<b>Range of values</b>	Formula One Racer
<b>Domain</b>	Grand Prix

Table 3.5: Table of Relation attribute: first Driver

<b>Name</b>	(has) pole Team
<b>Description</b>	In motor sport the pole position is the position at the inside of the front row at the start of a racing event. This position is typically given to the driver and the car (constructor team) with the best qualifying time in the trials before the race. This number-one qualifying team is referred to as the pole team. The relation “(has) pole Team” is a relation between the F1 team and Grand Prix.
<b>Range of values</b>	Formula One Team
<b>Domain</b>	Grand Prix

Table 3.6: Table of Relation attribute: pole Team

To finalize Task 3, the tables of instances should be listed. In practice,

duo to the large number of instances, the information about instance attribute and relations are directly provided in the implementation phase where all these data are extracted from the reliable sources and represented in RDF formats in a knowledge base. Figure 3.4 shows an example of table of instance which describes an instance of Formula One Racer (Sebastian Vettel).

Instance Name	Description	Attribution/value	Relation/value
Sebastian Vettel	He is a German racing driver, currently driving in Formula One for Scuderia Ferrari. He is a four-time Formula One World Champion, having won the championship in 2010, 2011, 2012, and 2013 with Red Bull Racing. He is among the most successful F1 drivers of all time. He is one of only four drivers to have won four or more drivers' titles. He is contracted to remain as a Formula One driver with Scuderia Ferrari until at least the end of 2017.	Poles=46	First win=2008ItalianGP
		Wins=42	Last win=2015SingaporeGP
		Championships=4	First race=2007UnitedStates
		Podiums=86	Last race=2016AbuDhabiGP
		Fastest Laps=28	
		Races=179	

Figure 3.4: Table of Instances: Sebastian Vettel

In Task 4, the goal is to build a preliminary conceptual model based on the gathered knowledge in the previous tasks of conceptualization activity. RDF statements (triples) and RDF graphs are the proposed techniques to formally structure and visualize this knowledge. As mentioned before, RDF statements are triples of the form “subject, predicate, object” to describe the resources; and RDF graphs are used to visualize RDF statements easily for human use. They consist pairs of nodes (subject and object) connected by an edge (predicate).

To do Task 4, we reused the RDF triples available in the documentation of DBpedia’s website, which were related to the knowledge gathered in data dictionary and other tables. We divided these triples into three parts and drew RDF graphs regarding to these parts.

In the first part, we considered the RDF triples about the part of the knowledge in Data Dictionary including Formula One Classes, instance relations and attributes. The resulted RDF graph from these triples is shown in Figure 3.5. This graph of DBpedia ontology in domain of Formula One, is called “preliminary conceptual model”, designed for Formula One domain.

In this RDF graph which is based on DBpedia ontology, the “subject” and “object” parts of an RDF triple are represented by the DBpedia resources- “class” in the ontology refers to the set of resources of a specific type- and “predicate” part which defines the property for the subject, is represented by the “attribute”

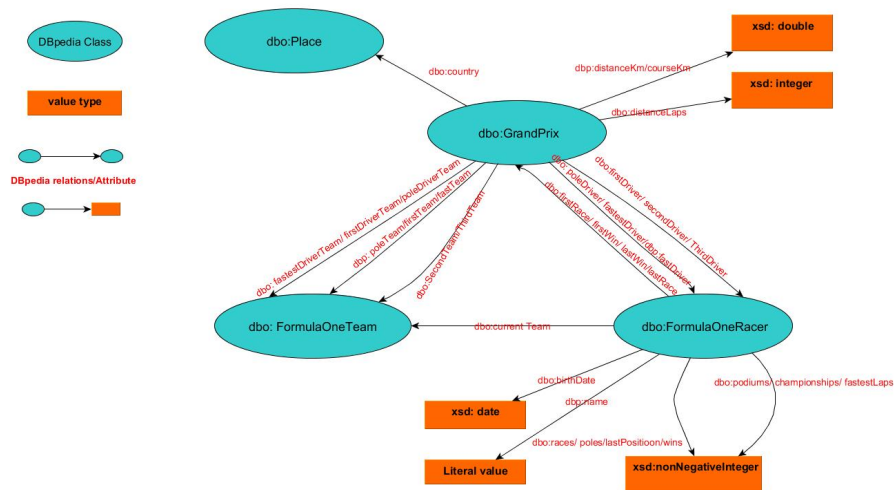


Figure 3.5: Preliminary conceptual model of Formula One domain based on DBpedia ontology

or “relation” in the ontology. The object part can be “literal” as well. All the parts of RDF triples should have a unique identifier (URI) except for the time when object is a literal and is not a resource. These URIs in the Figure 3.7 are abbreviated using “dbo” and “xsd” which are summarized form of these URIs:

- <http://www.dbpedia.org/ontology>
- <http://www.w3.org/2001/XMLSchema#>

In the second part, RDF statements about hierarchical structure of DBpedia’s classes, were considered to visualize. As can be seen in Figure 3.5, the main concepts (classes) from DBpedia ontology that are directly involved in Formula One domain: Formula One Racer, Grand Prix and Formula One Team, generally belong to a large subsuming hierarchy that is formed by 320 classes of DBpedia ontology and are described by 1,650 different properties [18]. Figure 3.6 depicts the hierarchical relations of the main Formula One classes with other classes of DBpedia ontology as well as sample properties that describe the instances of these classes.

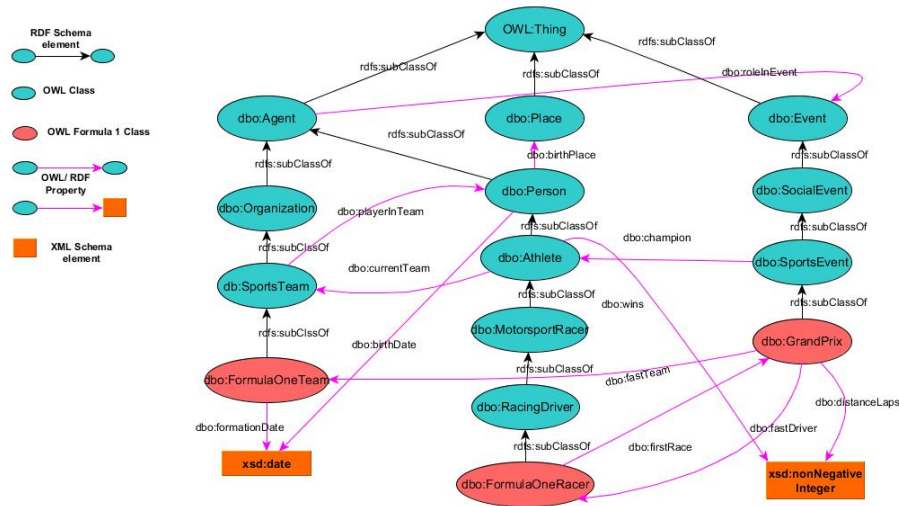


Figure 3.6: Snapshot of a part of DBpedia ontology with focus on Formula One classes

In the third part, the RDF triples to represent knowledge in instance level were chosen to visualize in an RDF graph. The sample instances of Formula One Classes as well as their attributes and relations' values are shown in Figure 3.7. Similar details that are provided during the previous task in table of Instances, such as type of the property, range, and domain, can be seen in this graph. These terminological knowledge (RDFS/OWL) can be seen in the middle and top parts of the graph and assertion knowledge (RDF) is expressed at the bottom part. The details about ontology language (RDFS elements) are provided in the next section (Implementation phase).

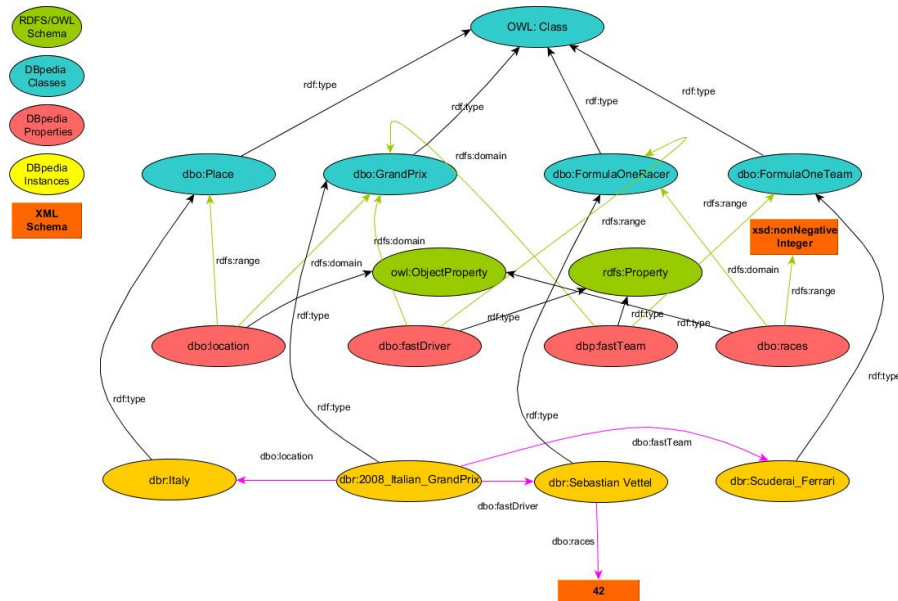


Figure 3.7: Snapshot of a part of DBpedia ontology with focus on Formula One instances

In the next step, the terminological knowledge represented in the preliminary model should be evaluated. The goal is to determine how much this model can fulfill the requirements specified in the specification phase?, which kinds of competency questions it cannot provide answer for? and what are the missing parts?

Reviewing the preliminary conceptual model based on DBpedia ontology and competency questions specified in the specification phase, led us to this conclusion that some questions cannot be directly or indirectly answered using this model. These questions are listed below:

- Questions about Grands Prix in which the round number of a race in a season, its exact location (Grand Prix circuit or the name of race track) and its exact date is questioned.
- Questions about the Formula One Racer (driver) that look for the information about the years that the athlete (driver) was active in Formula One races; about the teams that he participated in, during different seasons; about the points and the position that the driver gained in different seasons.
- Questions that are intended to find statistical information about one specific Formula One season like which driver or team was the champion of

that season, which Grands Prix, teams and drivers participated in that season?

- Statistical questions about a Formula One team like the number of wins, champion titles, fastest laps, pole positions and the drivers who had contract with them during different seasons.

To modify the current model to be able to answer these kind of questions, two classes for defining Formula One season, race circuit and a number of properties for different objects are considered to be added to this model.

To represent this complementary knowledge, we expressed them in a structured format, using RDF triples: <subject><predicate><object>. (For simplicity in writing, the prefixes in triples are ignored). Then they are visualized using RDF graph and added to the current model to build the optimized conceptual model. These proposed solutions in form of RDF triples, are listed below:

- Introducing “Formula One Season” and adding predicates to the instances of this class such as: “constructors Champion”, “drivers Champion”, “races”, “started In” and “season Date” to provide information about the champions, place, date and number of races in each season. The example triple below, introduces F1 Season, and shows the information about the instance “2010 Formula One Season “:

```
<F1-Season><type Of><F1-Class>
<2010-F1-Season><type Of><F1-Season>
<2010-F1-Season><constructors Champion><Red-Bull-Racing>
<2010-F1-Season><started In><Bahrain>
<2010-F1-Season><season Date> 2010
<2010-F1-Season><drivers Champion><Sebastian-Vettel>
<2010-F1-Season><races> 19
```

- Completing information about “Grand Prix” by introducing a new class named “Formula One Circuit” and adding several new predicates to all instances of “Grand Prix”. For example, the predicates that are listed in triples below for the instance “2011 Australian Grand Prix”, can provide information about the season, round number of GP, circuit name and the exact date and place of the race.

```
<Formula-One-Circuit><type Of><F1-Class>
<2011-Australian-Grand-Prix><part Of><2011-F1-Season>
<2011-Australian-Grand-Prix><race Of Season> 1
<2011-Australian-Grand-Prix><circuit><Melbourne-Grand-Prix-Circuit>
<Melbourne-Grand-Prix-Circuit><type Of><Formula-One-Circuit>
<2011-Australian-Grand-Prix><race Date> 27 March 2011
<2011-Australian-Grand-Prix><hosted In><Australia>
```



- Completing information about “Formula One Racer” class by adding several predicates to all instances of this class. For example, the predicates like “has Contract” and “has Records” for the driver “Sebastian Vettel” can provide statistical information about the driver’s membership in different teams during certain seasons and statistical information about all the records that the driver gained during different seasons, respectively. These records include number of races, wins, fastest Laps, poles, podiums, points, positions that a driver gained in a specific season. In addition, two more predicates were added to determine the first and last seasons each driver has participated in F1 races.

In the preliminary model, there are some predicates (attributes) with the same name of these records for the drivers, but these attributes provide the total number of the records during the whole Formula One Career of the driver. For instance, from the current model we can figure out how many times “Sebastian Vettel” was in podium during his whole F1 career, but we cannot figure out how many times he was in podium in a specific season. Triples below are some examples from the added part to ontology about the driver “Sebastian Vettel”.

```

<Sebastian-Vettel><has Contract><contract-123>
<contract-123><with Team><Toro-Rosso>
<contract-123><in Year> 2007
<Sebastian-Vettel><has Contract><contract-456>
<contract-456><with Team><Toro-Rosso>
<contract-456><in Year> 2008
<Sebastian-Vettel><has records><record-2010>
<record-2010><season><2010-F1-Season>
<record-2010><points> 256
<record-2010><races> 19
<record-2010><fastest Laps> 3
<record-2010><poles> 10
<record-2010><podiums> 10
<record-2010><wins> 5
<record-2010><position> 1
<Sebastian-Vettel><first Season><2007-F1-Season>
<Sebastian-Vettel><last Season><2017-F1-Season>

```

- Adding predicates to instances of “Formula One Team” class to provide information about the number of drivers/team’s champion titles, race victories, points, podiums, fastest Laps and pole positions which a particular team has gained as well as the first and latest entries of the team to the

F1 races, the first and the last wins of a winner team and the number of races in which it has participated in. The triples below provide these information about the team “Red Bull Racing”:

```
<Red-Bull-Racing ><first Entry><2005-Australian-Grand-Prix>
<Red-Bull-Racing ><last Entry><2017-Russian-Grand-Prix>
<Red-Bull-Racing ><drivers Championships Number> 4
<Red-Bull-Racing ><constructors Championships Number> 4
<Red-Bull-Racing ><races> 227
<Red-Bull-Racing ><wins> 52
<Red-Bull-Racing ><pole Positions> 58
<Red-Bull-Racing ><fastest Laps> 52
<Red-Bull-Racing ><points> 3557.5
<Red-Bull-Racing ><podiums> 135
<Red-Bull-Racing ><first Win><2009-Chinese-Grand-Prix>
<Red-Bull-Racing ><last Win><2016-Malaysian-Grand-Prix>
```

After applying all the modifications mentioned above, the preliminary model based on DBpedia ontology was extended to the new conceptual model (new ontology), designed specifically for domain of Formula One. Figure 3.8 depicts this optimized model.

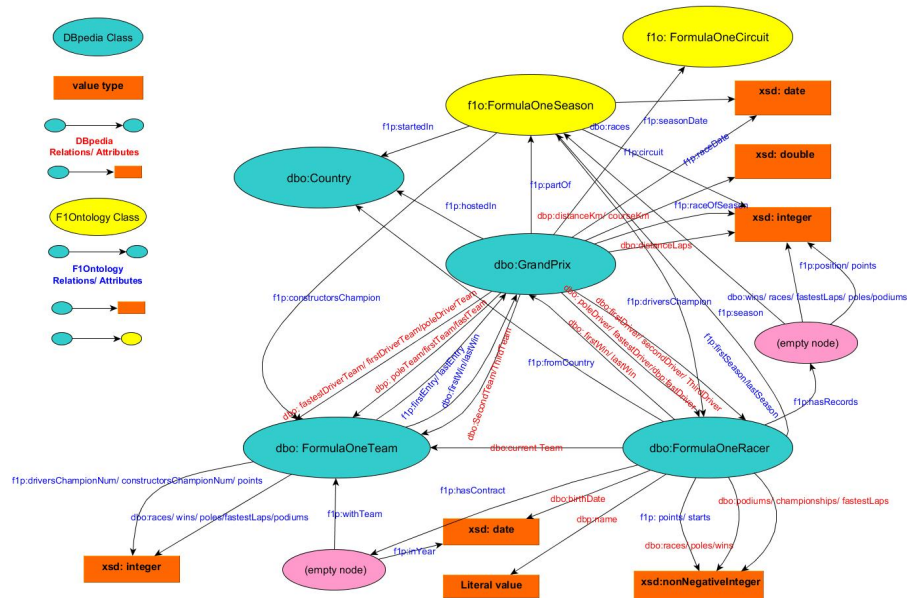


Figure 3.8: Conceptual model of Formula One domain based on new ontology

The empty (blank) nodes in the model represent resources which are anonymous. It means (real) URIs or literals are not assigned to them. According to the RDF standard, these blank nodes are only used as subjects or objects of RDF triples. They helped us to group sub-properties within a parent property. For example, we could define sub-properties (detailed records of each driver in each season) for the “has Records” predicate defined for an instance of “Racer” class.

## 3.2 Implementation of the ontology

In this step, we want to convert the conceptual model (F1 ontology) to a machine-readable format. In other words, all the specified knowledge in the conceptualization phase, must be collected completely and converted into a format that can be understandable for machines. Therefore, implementation activity consists of two main tasks: first, providing (collecting) data for the specified knowledge in conceptualization phase, second converting this knowledge in machine-readable format and storing it in an appropriate database.

### 3.2.1 Collecting data

In the first task, we should collect all the required data for the specified knowledge in our ontology. As mentioned before, in the new ontology, we have two parts of knowledge:

- The part that introduces the classes (entities), along with their subsuming hierarchies and define properties or relations for each entity with appropriate domain and range restrictions. This part is called “terminological knowledge” of the ontology (also known as schema) that add semantics to the data (RDF) with the help of an ontology language like RDFS; Resource Description Framework Schema. This knowledge is mainly visualized in the new conceptual model (Figure 3.8).
- The part that provides information about the instances of the classes in domain of Formula One, which is called “assertional knowledge” of the ontology where data about instances of classes namely F1 Racer, F1 Team, Grands Prix, F1 Circuit and F1 Season and Country is presented in standard data model and all the resources (instances) are given unique identifiers.

The following sections, introduces the resources where we collected data for each part separately:

**1. Collection of data for schema part:** by referring to the new conceptual model which demonstrates schema knowledge, for each entity in the model as well as all the attributes and relations among them, which belong to DBpedia ontology and are specified by red color in the figure, we collected RDF-data from

the associated path on the DBpedia official web site: “dbpedia.org/data/[name of entity or property].ntriples “

Figure 3.9 shows a snapshot of a DBpedia web page including N-triples about “GrandPrix” entity and the properties whose domains or ranges are defined by this entity. The highlighted lines, refer to the sample triples which were extracted by a script, to define “Grand Prix” entity and its related properties in our ontology.

Figure 3.9: Snapshot of a DBpedia web page including schema triples about Grand Prix entity

On the other hand, all the data related to defining the entities and properties of new ontology which were specified by blue color in the conceptual model, were gathered manually from the Wikipedia articles about Formula One.

**2. Collection of data for instances:** As Wikipedia is considered a huge information resource about Formula One, we selected this encyclopedia as the main source to collect data about instances of entities in Formula One. Except for “Grand Prix” instances whose information is driven directly from Dbpedia database, the data about all other instances are driven from Wikipedia tables. The following sections introduce data resources for defining the instances of classes in F1: Racer (Driver), Country, Season, Circuit, Team, and Grand Prix.

### 2.1 F1 Racer and Country

Figure 3.10 shows the Wikipedia table where we collected a list of drivers’ names, the countries where they come from along with the total records in

their F1 careers. The first and last season where they participated in F1 races, are also other related data that we collected about drivers (according to the ontology). All the fields in the table are used to extract values and have been pointed by the arrows.

The screenshot shows a Wikipedia table titled "List of Formula One drivers". The table has 11 columns: Name, Country, Seasons, Championships, Entries, Starts, Poles, Wins, Podiums, Fastest laps, and Points. Red arrows point to the following columns: Name, Country, Seasons, Championships, Entries, Starts, Poles, Wins, Podiums, Fastest laps, and Points.

Name	Country	Seasons	Championships	Entries	Starts	Poles	Wins	Podiums	Fastest laps	Points <sup>[note]</sup>
Carlo Abate	<span><span></span></span> Italy	1962–1963	0	2	0	0	0	0	0	0
George Abecassis	<span><span></span></span> United Kingdom	1951–1952	0	2	2	0	0	0	0	0
Kenny Acheson	<span><span></span></span> United Kingdom	1983, 1985	0	10	3	0	0	0	0	0
Andrea de Adamich	<span><span></span></span> Italy	1968, 1970–1973	0	36	30	0	0	0	0	6
Philippe Adams	<span><span></span></span> Belgium	1994	0	2	2	0	0	0	0	0
Walt Ader	<span><span></span></span> United States	1950	0	1 <sup>[1]</sup>	1	0	0	0	0	0
Kurt Adloff	<span><span></span></span> Germany	1953	0	1	1	0	0	0	0	0
Fred Agabashian	<span><span></span></span> United States	1950–1957	0	9 <sup>[1]</sup>	8	1	0	0	0	1.5
Kurt Ahrens, Jr.	<span><span></span></span> Germany	1968	0	1	1	0	0	0	0	0
Christijan Albers	<span><span></span></span> Netherlands	2005–2007	0	46	46	0	0	0	0	4
Michele Alboreto	<span><span></span></span> Italy	1981–1994	0	215	194	2	5	23	5	186.5
Jean Alesi	<span><span></span></span> France	1989–2001	0	202	201	2	1	32	4	241

Figure 3.10: Snapshot of a Wikipedia web page including names, countries and total records of drivers

The information about the current team of active drivers (about 98 drivers) is driven from the table shown in Figure 3.11. The values for the first and last Grands Prix where drivers won, are provided from the table shown in Figure 3.12. The arrows point at the selected fields for data collection.

Formula One driver numbers [\[ edit \]](#)

The following lists all Formula One driver numbers currently claimed, as of the 2017 season:

No.	Driver	Current team	Earliest year available	Notes
1	Reserved for Champion	N/A	2018	<i>The 2016 Champion is not competing in 2017</i>
2	<span><span><span></span></span><span> </span></span> Stoffel Vandoorne	<span><span><span></span></span><span> </span></span> McLaren	2020	
3	<span><span><span></span></span><span> </span></span> Daniel Ricciardo	<span><span><span></span></span><span> </span></span> Red Bull	2020	
5	<span><span><span></span></span><span> </span></span> Sebastian Vettel	<span><span><span></span></span><span> </span></span> Ferrari	2020	
6	<span><span><span></span></span><span> </span></span> Nico Rosberg	<i>Last raced in 2016</i>	2019	
7	<span><span><span></span></span><span> </span></span> Kimi Räikkönen	<span><span><span></span></span><span> </span></span> Ferrari	2020	
8	<span><span><span></span></span><span> </span></span> Romain Grosjean	<span><span><span></span></span><span> </span></span> Haas	2020	
9	<span><span><span></span></span><span> </span></span> Marcus Ericsson	<span><span><span></span></span><span> </span></span> Sauber	2020	
11	<span><span><span></span></span><span> </span></span> Sergio Pérez	<span><span><span></span></span><span> </span></span> Force India	2020	
12	<span><span><span></span></span><span> </span></span> Felipe Nasr	<i>Last raced in 2016</i>	2019	
13	<span><span><span></span></span><span> </span></span> Pastor Maldonado	<i>Last raced in 2015</i>	2018	
14	<span><span><span></span></span><span> </span></span> Fernando Alonso	<span><span><span></span></span><span> </span></span> McLaren	2020	
17	<span><span><span></span></span><span> </span></span> Jules Bianchi	<i>Last raced in 2014</i>	N/A	<i>Retired by FIA following Bianchi's death in 2015</i>
18	<span><span><span></span></span><span> </span></span> Lance Stroll	<span><span><span></span></span><span> </span></span> Williams	2020	
19	<span><span><span></span></span><span> </span></span> Felipe Massa	<span><span><span></span></span><span> </span></span> Williams	2020	
20	<span><span><span></span></span><span> </span></span> Kevin Magnussen	<span><span><span></span></span><span> </span></span> Haas	2020	
21	<span><span><span></span></span><span> </span></span> Esteban Gutiérrez	<i>Last raced in 2016</i>	2019	

Figure 3.11: Snapshot of a Wikipedia web page including current teams of drivers

Secure | [https://en.wikipedia.org/wiki/List\\_of\\_Formula\\_One\\_Grand\\_Prix\\_winners](https://en.wikipedia.org/wiki/List_of_Formula_One_Grand_Prix_winners)

**Formula One Grand Prix Winners**

Rank	Country	Driver	Wins	Seasons active	First win	Last win
1	<span><span><span></span></span><span> </span></span> Germany	<b>Michael Schumacher</b>	91	1991–2006 2010–2012	1992 Belgian Grand Prix	2006 Chinese Grand Prix
2	<span><span><span></span></span><span> </span></span> United Kingdom	<b>Lewis Hamilton</b> †	56	2007–	2007 Canadian Grand Prix	2017 Canadian Grand Prix
3	<span><span><span></span></span><span> </span></span> France	<b>Alain Prost</b>	51	1980–1991 1993	1981 French Grand Prix	1993 German Grand Prix
4	<span><span><span></span></span><span> </span></span> Germany	<b>Sebastian Vettel</b> †	45	2007–	2008 Italian Grand Prix	2017 Monaco Grand Prix
5	<span><span><span></span></span><span> </span></span> Brazil	<b>Ayrton Senna</b>	41	1984–1994	1985 Portuguese Grand Prix	1993 Australian Grand Prix
6	<span><span><span></span></span><span> </span></span> Spain	<b>Fernando Alonso</b> †	32	2001 2003–	2003 Hungarian Grand Prix	2013 Spanish Grand Prix
7	<span><span><span></span></span><span> </span></span> United Kingdom	<b>Nigel Mansell</b>	31	1980–1992 1994–1995	1985 European Grand Prix	1994 Australian Grand Prix
8	<span><span><span></span></span><span> </span></span> United Kingdom	<b>Jackie Stewart</b>	27	1965–1973	1965 Italian Grand Prix	1973 German Grand Prix
9	<span><span><span></span></span><span> </span></span> United Kingdom	<b>Jim Clark</b>	25	1960–1968	1962 Belgian Grand Prix	1968 South African Grand Prix

Figure 3.12: Snapshot of a Wikipedia web page including first and last wins of GP winners (drivers)

In addition, the detailed records of all drivers in every season are collected from wiki-pages of all Formula One seasons, in the tables where drivers' standings are recorded. The big red-line arrow, points at all the fields that were used for collecting information. Driver's name and his points are directly extracted

from the table, but values of other properties like number of podiums, poles, fastest laps and wins are extracted indirectly from the table after a process is performed on the cells below the columns of the countries 'names. The content of each cell represents the value, background color and font formats.

The numbers/ letters and the background color represent the result (position) of each participant driver in a race; "Bold" and "Italic" fonts refer to this fact that if the driver has been a pole sitter or a fast driver in a specific race. The yellow-line arrows point at this key information inside and below the "Key" Table shown in Figure 3.13. The figure shows a sample table from season 2015 with the data explained above.

In the event of a tie, a count-back system was used as a tie-breaker, with a driver's best result used to decide the standings.<sup>[N 1]</sup>

Pos.	Driver	Races																Points	Key				
		AUS	MAL	CHN	BHR	ESP	MON	CAN	AUT	GBR	HUN	BEL	ITA	SIN	JPN	RUS	USA		MEX	BRA	ABU	Colour	Result
1	Lewis Hamilton	1	2	7	1	2	3	1	2	1	6	1	1	Ret	1	1	1	2	2	2	381	Gold	Winner
2	Nico Rosberg	2	3	2	3	1	1	2	1	2	8	2	17†	4	2	Ret	2	1	1	1	322	Silver	2nd place
3	Sebastian Vettel	3	1	3	5	3	2	5	4	3	1	12†	2	1	3	2	3	Ret	3	4	278	Bronze	3rd place
4	Kimi Räikkönen	Ret	4	4	2	5	6	4	Ret	8	Ret	7	5	3	4	8	Ret	Ret	4	3	150	Green	Other points position
5	Valtteri Bottas	DNS	5	6	4	4	14	3	5	5	13	9	4	5	5	12†	Ret	3	5	13	136	Blue	Other classified position
6	Felipe Massa	4	6	5	10	6	15	6	3	4	12	6	3	Ret	17	4	Ret	6	DSQ	8	121	Blue	Not classified, finished (NC)
7	Daniil Kvyat	DNS	9	Ret	9	10	4	9	12	6	2	4	10	6	13	5	Ret	4	7	10	95	Purple	Not classified, retired (Ret)
8	Daniel Ricciardo	6	10	9	6	7	5	13	10	Ret	3	Ret	8	2	15	15†	10	5	11	6	92	Red	Did not qualify (DNQ)
9	Sergio Pérez	10	13	11	8	13	7	11	9	9	Ret	5	6	7	12	3	5	8	12	5	78	Red	Did not pre-qualify (DNPO)
10	Nico Hülkenberg	7	14	Ret	13	15	11	8	6	7	Ret	DNS	7	Ret	6	Ret	Ret	7	6	7	58	Black	Disqualified (DSQ)
11	Romain Grosjean	Ret	11	7	7	8	12	10	Ret	Ret	7	3	Ret	13†	7	Ret	Ret	10	8	9	51	White	Did not start (DNS)
12	Max Verstappen	Ret	7	17†	Ret	11	Ret	15	8	Ret	4	8	12	8	9	10	4	9	9	16	49	White	Race canceled (C)
13	Felipe Nasr	5	12	8	12	12	9	16	11	DNS	11	11	13	10	20†	6	9	Ret	13	15	27	Blank	Did not practice (DNP)
14	Pastor Maldonado	Ret	Ret	Ret	15	Ret	Ret	7	7	Ret	14	Ret	Ret	12	8	7	8	11	10	Ret	27	Blank	Excluded (EX)
15	Carlos Sainz Jr.	9	8	13	Ret	9	10	12	Ret	Ret	Ret	Ret	11	9	10	Ret	7	13	Ret	11	18	Blank	Did not arrive (DNA)
16	Jenson Button	11	Ret	14	DNS	16	8	Ret	Ret	Ret	9	14	14	Ret	16	9	6	14	14	12	16	Blank	Withdrawn (WD)
17	Fernando Alonso	Ret	12	11	Ret	Ret	Ret	Ret	Ret	10	5	13	18†	Ret	11	11	11	Ret	15	17	11	Bold	Pole position
18	Marcus Ericsson	8	Ret	10	14	14	13	14	13	11	10	10	9	11	14	Ret	Ret	12	16	14	9	Italics	Fastest lap

Figure 3.13: Snapshot of a Wikipedia web page including records of drivers in season 2015

Finally, the information about the teams that each driver had contract with, are collected from wiki-tables of F1 seasons where data about participant constructors (teams) and drivers are recorded. A sample table in "Season 2014" is shown in Figure 3.14. The arrows in the figure refer to the selected fields for extraction of values.

Teams and drivers [\[ edit \]](#)

The following teams and drivers took part in the 2014 Formula One World Championship.<sup>[2]</sup>

Entrant	Constructor	Chassis	Power unit	Tyre	No.	Race drivers	Rounds	No.	Free Practice drivers
 Scuderia Ferrari	Ferrari	SF15-T	Ferrari 059/4 <sup>[3]</sup>		5 7	 Sebastian Vettel  Kimi Räikkönen	All All		N/A
 Sahara Force India F1 Team	Force India-Mercedes	VJM08 VJM08B <sup>[4]</sup>	Mercedes PU106B Hybrid		11 27	 Sergio Pérez  Nico Hülkenberg	All All		N/A
 Lotus F1 Team	Lotus-Mercedes	E23 Hybrid	Mercedes PU106B Hybrid		8 13	 Romain Grosjean  Pastor Maldonado	All All	30	 Jolyon Palmer
 Manor Marussia F1 Team	Marussia-Ferrari	MR03b <sup>[5]</sup>	Ferrari 059/3 <sup>[7]</sup>		28 98 53	 Will Stevens  Roberto Merhi  Alexander Rossi	All <sup>2</sup> 1–12, 15, 19 13–14, 16–18	42	 Fabio Leimer
 McLaren Honda	McLaren-Honda	MP4-30	Honda RA615H		20 14 22	 Kevin Magnussen  Fernando Alonso  Jenson Button	1 2–19 All		N/A
 Mercedes AMG Petronas F1 Team	Mercedes	F1 W06 Hybrid	Mercedes PU106B Hybrid		6 44	 Nico Rosberg  Lewis Hamilton	All All		N/A
 Infiniti Red Bull Racing	Red Bull-Renault	RB11	Renault Energy F1-2015		3 26	 Daniel Ricciardo  Daniil Kvyat	All All		N/A
 Sauber F1 Team	Sauber-Ferrari	C34	Ferrari 059/4 <sup>[3]</sup>		9 12	 Marcus Ericsson  Felipe Nasr	All All	36	 Raffaele Marciello
 Scuderia Toro Rosso	Toro Rosso-Renault	STR10	Renault Energy F1-2015		33 55	 Max Verstappen  Daniel Ricciardo	All All		N/A

Figure 3.14: Snapshot of a Wikipedia web page including participant constructors and drivers in season 2014

## 2.2 F1 Season and Circuit

Figure 3.15 shows a part of a Wikipedia page about a specific season (2012 F1 season) that are used for extraction of necessary data for instances of “F1 Season” and “F1 Circuit” classes.

The table in the figure provides information about start date, round number, and circuit’s name of each instance of Grand Prix in that season. In addition, list of champions (teams and drivers) who gained the first position after completing a F1 season, are provided from the table shown in Figure 3.16.





Secure | [https://en.wikipedia.org/wiki/2012\\_Formula\\_One\\_season](https://en.wikipedia.org/wiki/2012_Formula_One_season)

Season calendar [ edit ]

Round	Grand Prix	Circuit	Date
1	Australian Grand Prix	Melbourne Grand Prix Circuit, Melbourne	18 March
2	Malaysian Grand Prix	Sepang International Circuit, Kuala Lumpur	25 March
3	Chinese Grand Prix	Shanghai International Circuit, Shanghai	15 April
4	Bahrain Grand Prix	Bahrain International Circuit, Sakhir	22 April
5	Spanish Grand Prix	Circuit de Catalunya, Montmeló	13 May
6	Monaco Grand Prix	Circuit de Monaco, Monte Carlo	27 May
7	Canadian Grand Prix	Circuit Gilles Villeneuve, Montreal	10 June
8	European Grand Prix	Valencia Street Circuit, Valencia	24 June
9	British Grand Prix	Silverstone Circuit, Silverstone	8 July
10	German Grand Prix	Hockenheimring, Hockenheim	22 July
11	Hungarian Grand Prix	Hungaroring, Budapest	29 July
12	Belgian Grand Prix	Circuit de Spa-Francorchamps, Stavelot	2 September
13	Italian Grand Prix	Autodromo Nazionale Monza, Monza	9 September
14	Singapore Grand Prix	Marina Bay Street Circuit, Singapore	23 September
15	Japanese Grand Prix	Suzuka Circuit, Mie	7 October
16	Korean Grand Prix	Korea International Circuit, Yeongam	14 October
17	Indian Grand Prix	Buddh International Circuit, Greater Noida	28 October

Figure 3.15: Snapshot of a Wikipedia web page including general info. about season 2012

Secure | [https://en.wikipedia.org/wiki/List\\_of\\_Formula\\_One\\_seasons](https://en.wikipedia.org/wiki/List_of_Formula_One_seasons)

List of seasons [edit]

Season	Races	Drivers' champion (team)	Constructors' champion
1950	7	<span><span></span></span> Nino Farina (Alfa Romeo)	<i>Not awarded</i>
1951	8	<span><span></span></span> Juan Manuel Fangio (Alfa Romeo)	<i>Not awarded</i>
1952	8	<span><span></span></span> Alberto Ascari (Ferrari)	<i>Not awarded</i>
1953	9	<span><span></span></span> Alberto Ascari (Ferrari)	<i>Not awarded</i>
1954	9	<span><span></span></span> Juan Manuel Fangio (Maserati/Mercedes)	<i>Not awarded</i>
1955	7	<span><span></span></span> Juan Manuel Fangio (Mercedes)	<i>Not awarded</i>
1956	8	<span><span></span></span> Juan Manuel Fangio (Ferrari)	<i>Not awarded</i>
1957	8	<span><span></span></span> Juan Manuel Fangio (Maserati)	<i>Not awarded</i>
1958	11	<span><span></span></span> Mike Hawthorn (Ferrari)	<span><span></span></span> Vanwall
1959	9	<span><span></span></span> Jack Brabham (Cooper)	<span><span></span></span> Cooper-Climax
1960	10	<span><span></span></span> Jack Brabham (Cooper)	<span><span></span></span> Cooper-Climax
1961	8	<span><span></span></span> Phil Hill (Ferrari)	<span><span></span></span> Ferrari
1962	9	<span><span></span></span> Graham Hill (BRM)	<span><span></span></span> BRM
1963	10	<span><span></span></span> Jim Clark (Lotus)	<span><span></span></span> Lotus-Climax
1964	10	<span><span></span></span> John Surtees (Ferrari)	<span><span></span></span> Ferrari

Figure 3.16: Snapshot of a Wikipedia web page including champions of all seasons

### 2.3 F1 Team

We collected the required statistics about different constructors, according to the ontology, from a Wikipedia table named “former constructors’ statistics”. Figure 3.17 depicts the selected fields of this table (pointed by red-line arrows). Moreover, the statistics of “2017 constructors” were collected from a similar table (2017constructors’ statistics).

Former constructors' statistics [edit]

Key: Races Entered = Number of individual races entered; Races Started = Number of individual races started; Drivers = Number of drivers; Total Entries = Total number of race entries; Wins = Number of races won; Points = Number of Constructors Championship points scored; Poles = Number of pole positions; FL = Number of flag laps; Podiums = Number of podium finishes; WCC = Constructors' Championships won; WDC = Drivers' Championships won.

Constructor	Country	Seasons	Races Entered	Races Started	Drivers	Total Entries	Wins	Points	Poles	FL	Podiums	WCC	WDC	First Grand Prix	Last Grand Prix
Alex von Falkenhausen Motorenbau	<span><span><span></span></span><span> </span></span> GER	1952–1953 <sup>[6]</sup>	4	4	6	9	0	n/a	0	0	0	n/a	0	1952 Swiss	1953 Italian
Automobiles Gonfaronnaises Sportives	<span><span><span></span></span><span> </span></span> FRA	1986–1991	80	32	10	124	0	2	0	0	0	0	0	1986 Italian	1991 Monaco
Alfa Romeo	<span><span><span></span></span><span> </span></span> ITA	1950–1951 1979–1985	110	110	18	237	10	50 <sup>[6]</sup>	12	14	26	0 <sup>[7]</sup>	2	1950 British	1985 Australian
Alta	<span><span><span></span></span><span> </span></span> GBR	1950–1952 <sup>[6]</sup>	5	5	4	6	0	n/a	0	0	0	n/a	0	1950 British	1952 British
Amon	<span><span><span></span></span><span> </span></span> ITA	1974	4	1	2	4	0	0	0	0	0	0	0	1974 Spanish	1974 Italian
Andrea Moda	<span><span><span></span></span><span> </span></span> ITA	1992	12	1	4	16	0	0	0	0	0	0	0	1992 S. African	1992 Italian

Figure 3.17: Snapshot of a Wikipedia web page including total records of teams

Similar to the case of drivers, the list of first and last races where constructors (teams) won the Grands Prix are gathered from the table shown in Figure 3.18.

https://en.wikipedia.org/wiki/List\_of\_Formula\_One\_Grand\_Prix\_winners\_(constructors)

Formula One World Champion

Rank	Country	Constructor	Wins	Active	First win	Last win	Notes
1	<span><span><span></span></span><span> </span></span> ITA	<i>Ferrari</i>	227	1950–	1951 British Grand Prix	2017 Monaco Grand Prix	[4]
2	<span><span><span></span></span><span> </span></span> GBR	<i>McLaren</i>	182	1966–	1968 Belgian Grand Prix	2012 Brazilian Grand Prix	[5]
3	<span><span><span></span></span><span> </span></span> GBR	<i>Williams</i>	114	1975–	1979 British Grand Prix	2012 Spanish Grand Prix	[6]
4	<span><span><span></span></span><span> </span></span> GBR	<i>Team Lotus</i>	79	1958–1994	1960 Monaco Grand Prix	1987 Detroit Grand Prix	[3]
5	<span><span><span></span></span><span> </span></span> GER	<i>Mercedes</i>	68	1954–1955 2010–	1954 French Grand Prix	2017 Canadian Grand Prix	[3]
6	<span><span><span></span></span><span> </span></span> AUT	<i>Red Bull</i>	52	2005–	2009 Chinese Grand Prix	2016 Malaysian Grand Prix	[7]
7	<span><span><span></span></span><span> </span></span> GBR	<i>Brabham</i>	35	1962–1992	1964 French Grand Prix	1985 French Grand Prix	[3]
7	<span><span><span></span></span><span> </span></span> FRA	<i>Renault</i>	35	1977–1985 2002–2011 2016–	1979 French Grand Prix	2008 Japanese Grand Prix	[8]
9	<span><span><span></span></span><span> </span></span> GBR <span><span><span></span></span><span> </span></span> ITA <sup>[A]</sup>	<i>Benetton</i>	27	1986–2001	1986 Mexican Grand Prix	1997 German Grand Prix	[3][9]
10	<span><span><span></span></span><span> </span></span> GBR	<i>Tyrrell</i>	23	1970–1998	1971 Spanish Grand Prix	1983 Detroit Grand Prix	[3]
11	<span><span><span></span></span><span> </span></span> GBR	<i>BRM</i>	17	1951 1956–1977	1959 Dutch Grand Prix	1972 Monaco Grand Prix	[3]
12	<span><span><span></span></span><span> </span></span> GBR	<i>Cooper</i>	16	1950 1952–1969	1958 Argentine Grand Prix	1967 South African Grand Prix	[3]

Figure 3.18: Snapshot of a Wikipedia web page including first and last wins of GP winners (teams)

## 2.4 Grand Prix

Like collection of data for schema knowledge, all the required statistics according to the new ontology about instances of Grands Prix are gathered directly from the Dbpedia knowledge base which is stored in the Dbpedia official website: “www.dbpedia.org”. The reason behind is that, Dbpedia extracts information from Wikipedia which is the main source of data for our ontology and converts them to structured format (like RDF statements) which is the target format of our data to be stored in a database.

Figure 3.19 shows a snapshot of a DBpedia web page including N-triples that define the instance of “2014 Canadian Grand Prix” in terms of specified properties and relations in our ontology. The highlighted lines, refer to the sample triples which include the required data about this instance.

```

<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/ontology/poleDriver><http://dbpedia.org/resource/Nico_Rosberg>
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/ontology/secondDriver><http://dbpedia.org/resource/Nico_Rosberg> .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://www.w3.org/1999/02/22-rdf-syntax-ns#type><http://schema.org/Event> .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://www.w3.org/2000/01/rdf-schema#comment>"De Grand Prix Formule 1 van Canada 2014
werd gehouden op 8 juni 2014 op het Circuit Gilles Villeneuve. Het was de zevende race van het kampioenschap."@nl .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/property/firstDriver><http://dbpedia.org/resource/Daniel_Ricciardo> .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://www.w3.org/2000/01/rdf-schema#label> "2014 Canadian Grand Prix"@en .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://www.w3.org/2000/01/rdf-schema#label>
"2014\u5E74\u30AB\u30C8\u30C0\u30B0\u30E9\u30F3\u30D7\u30EA"@ja .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://www.w3.org/2002/07/owl#sameAs>
<http://pt.dbpedia.org/resource/Grande_Pr\u00EAmio_do_Canad\u00E1_de_2014_(F\u00F3rmula_1)> .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/ontology/thumbnaill>
<http://commons.wikimedia.org/wiki/Special:FilePath/Circuit_Gilles_Villeneuve.svg?
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/ontology/GrandPrix/course>"4.361"^^<http://dbpedia.org/datatype/kilometr
e> .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://www.w3.org/2000/01/rdf-schema#label> "2014 Canadian Grand Prix"@en .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/property/officialName> "Formula 1 Grand Prix du Canada 2014"@en .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://www.w3.org/2002/07/owl#sameAs><http://nl.dbpedia.org/resource/Grand_Prix_Formule
_1_van_Canada_2014> .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/ontology/firstDriverTeam><http://dbpedia.org/resource/Red_Bull_Racing> .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/ontology/firstDriver><http://dbpedia.org/resource/Daniel_Ricciardo> .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/ontology/course>"4361.32"^^<http://www.w3.org/2001/XMLSchema#double
> .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/property/fastTeam><http://dbpedia.org/resource/Williams_Grand_Prix_Engi
neering> .
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/property/courseMi>"2.71"^^<http://www.w3.org/2001/XMLSchema#double>
.
<http://dbpedia.org/resource/2014_Canadian_Grand_Prix><http://dbpedia.org/ontology/distanceLaps>
"70"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger> .

```

Figure 3.19: Snapshot of a Dbpedia web page including sample triples about an instance of GP for extraction

### 3.2.2 Developing a knowledge base

In this step, we want to convert the collected data from the previous step to RDF-statements and store them in an appropriate database. For this purpose, several Python scripts are used to collect data for both schema and instances, from the mentioned resources (Wikipedia HTML tables and DBpedia pages). These scripts after extraction of the required data, convert the non-triples data to RDF statements. As mentioned before, there are several serialization formats for storing RDF statements in electronic documents (e.g. Turtle, N-Triples, RDF-XML, etc.). In this study, we used “N-Triples” standard which is very

simple, easy to parse and has a line-based format. In the final step, all the provided triples are saved in N3-files and stored in a database to construct a knowledge base. This process is summarized in Figure 3.20.

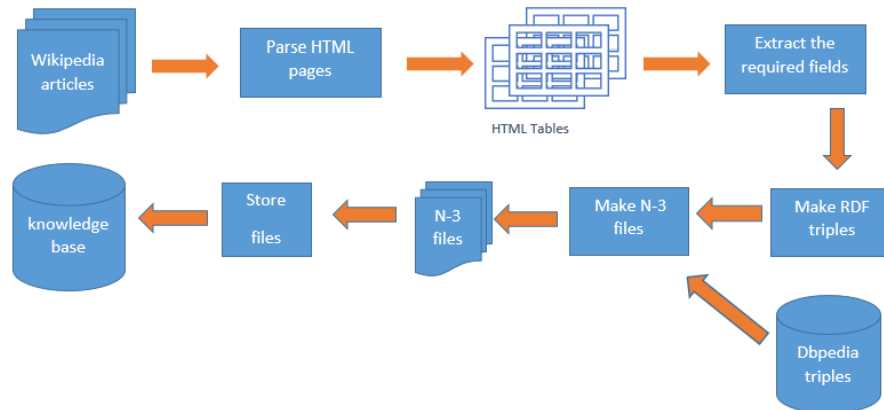



Figure 3.20: Process of making RDF triples and storing in a database

The database that we used to store triples is a kind of graph database known as “Ontotext GrapDB”. GraphDB is a Database engine, based fully on Semantic Web standards from W3C: RDF, RDFS, OWL and SPARQL. For this study, free version of GraphDB was used for storage of data which is a database engine for small projects.

To easy access to the RDF triples, we stored each part in a separate path (directory) of the knowledge base. The schema part was stored in a directory with the same name “schema” and the triples about instances of the F1 classes were stored in their corresponding directories namely “seasons”, “drivers”, “teams”, “Grandsprix”, “circuits” and “countries”.

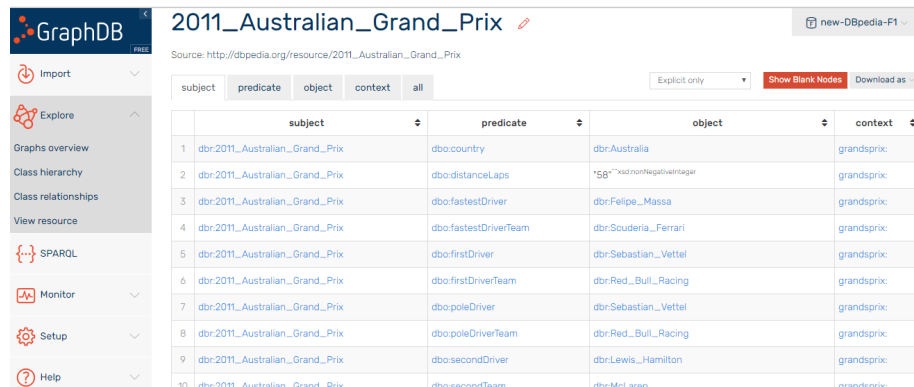
A snapshot of RDF N-Triples stored in the “schema” and “Grandsprix” directories of “Ontotext GrapDB”, are shown in the Figures 3.21 and 3.22 respectively. The first one, shows data about the class of “Sports Event” which is super class of “Grand Prix” in DBpedia ontology along with some related properties to “Formula One Racer” and “Grand Prix” classes and their restrictions.



	subject	predicate	object	context
27	dbo:SportsEvent	rdfs:subClassOf	dbo:SocietalEvent	http://newF1-ontolo
28	dbo:SportsTeam	rdf:type	owl:Class	http://newF1-ontology/schema
29	dbo:SportsTeam	rdfs:subClassOf	dbo:Organisation	http://newF1-ontology/schema
30	dbo:birthDate	rdf:type	rdf:Property	http://newF1-ontology/schema
31	dbo:birthDate	rdfs:domain	dbo:Person	http://newF1-ontology/schema
32	dbo:birthDate	rdfs:range	xsd:date	http://newF1-ontology/schema
33	dbo:birthPlace	rdf:type	rdf:Property	http://newF1-ontology/schema
34	dbo:birthPlace	rdfs:domain	dbo:Person	http://newF1-ontology/schema
35	dbo:birthPlace	rdfs:range	dbo:Place	http://newF1-ontology/schema
36	dbo:championships	rdf:type	rdf:Property	http://newF1-ontology/schema
37	dbo:championships	rdfs:domain	dbo:FormulaOneRacer	http://newF1-ontology/schema
38	dbo:championships	rdfs:range	xsd:nonNegativeInteger	http://newF1-ontology/schema
39	dbo:country	rdf:type	rdf:Property	http://newF1-ontology/schema
40	dbo:country	rdfs:domain	dbo:GrandPrix	http://newF1-ontology/schema

Figure 3.21: Snapshot of GraphDB including schema triples

Figure 3.22, represents triples about one instance of the class “Grand Prix” namely “2011 Australian Grand prix”. As can be seen, for the given resource (subject), all the properties below the “predicate” column, are given values below the “object” column which are either entities (resources with URIs) or literals.



	subject	predicate	object	context
1	dbr:2011_Australian_Grand_Prix	dbo:country	dbr:Australia	grandsprix
2	dbr:2011_Australian_Grand_Prix	dbo:distanceLaps	*58**xsd:nonNegativeInteger	grandsprix
3	dbr:2011_Australian_Grand_Prix	dbo:fastestDriver	dbr:Felipe_Massa	grandsprix
4	dbr:2011_Australian_Grand_Prix	dbo:fastestDriverTeam	dbr:Scuderia_Ferrari	grandsprix
5	dbr:2011_Australian_Grand_Prix	dbo:firstDriver	dbr:Sebastian_Vettel	grandsprix
6	dbr:2011_Australian_Grand_Prix	dbo:firstDriverTeam	dbr:Red_Bull_Racing	grandsprix
7	dbr:2011_Australian_Grand_Prix	dbo:poleDriver	dbr:Sebastian_Vettel	grandsprix
8	dbr:2011_Australian_Grand_Prix	dbo:poleDriverTeam	dbr:Red_Bull_Racing	grandsprix
9	dbr:2011_Australian_Grand_Prix	dbo:secondDriver	dbr:Lewis_Hamilton	grandsprix
10	dbr:2011_Australian_Grand_Prix	dbo:secondDriverTeam	dbr:Mercedes	grandsprix

Figure 3.22: Snapshot of GraphDB including triples of an instance of Grand prix

### 3.3 Extraction of answers

In this section, referring to the general model of QA system pictured in Chapter one (Fig 1.1); we present an overview of how the ontology and the knowledge base in online and offline stages, provide the QA system with necessary knowledge to return the result to the user.

Using the user interface, the question in natural language is sent to the “Question Processing” module. Referring to Blaauw’s report, the overall goal of this module is to interpret the NL question and convert it into a SPARQL query which is used for retrieving data in the knowledge base. The “Question Processing” module uses well known methods for NLP such as Part-of-speech tagging, chunking and lexical lookups. It also recognizes and maps "Named Entities" like people or organizations based on defined resources and phrases in the ontology. Finally, it generates queries through a greedy approach and selects the one which yield the expected correct result and send it to “Answer Extraction” module.

In this module, the generated query is searched against the prepared knowledge base and extracted answer is sent to the “Answer Preparation” component where the RDF-answer is converted to exact answer in NL format and if the RDF-answer is a kind of resource which has a specific URI like a driver’s name, the “label” of the resource is retrieved and is prepared as a linkable text to the “http” address of the resource in Wikipedia.

The answer is finally returned to the user via the “GUI” module as well as all the data that are used for processing the answer in a “web-form” structure. This processed data consists of different parts corresponds to the outputs of several components in the pipeline of "Question Processing" like the mapped phrases, recognized entities and SPARQL queries.

### 3.4 Implementing a prototype

The following section, presents a working prototype for the system that is implemented to represent all the functionality of the QA system. For this purpose, all the QA modules, are implemented through a software program which performs all the tasks for processing the questions and generation of the results.

In addition, the program provides a web interface, consisting of one web page with a search box, where the user can enter the question and receives the answer. This interface also has a simple possibility for fault handling through checking “empty input”.

The framework of prototype is constructed using Flask version 0.12, a micro web framework written in Python. With Flask, both the back-end (modules for providing answer) and the front-end (web interface) of the program is constructed through a combination of python code and common HTML syntax, where it is possible to include python code inside the HTML code. Figure 3.23 shows a picture of user interface while typing a question.

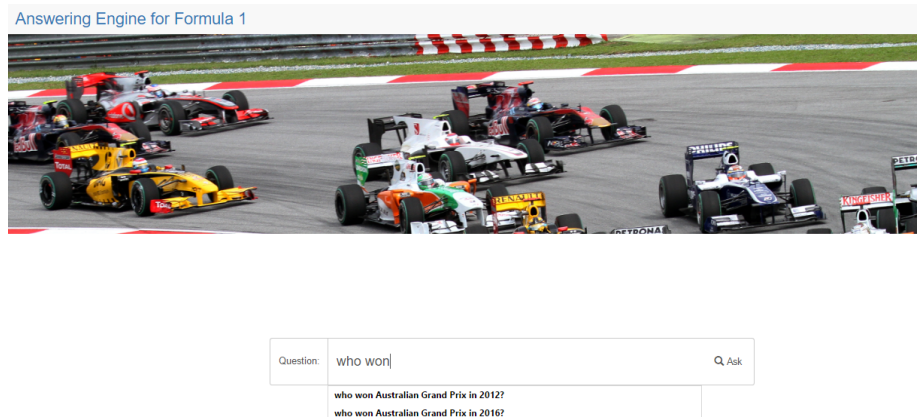


Figure 3.23: User interface of the QA system while typing a question

The user is free to ask his/ her own questions; by clicking on “Ask” button or choose a random question out of a prepared list from the system by clicking on a “Refresh” button. All the questions in this list are answered correctly by the system and have been checked before. The usability of the sample questions is to provide users with ideas and tips about what type of questions usually can be asked from the system. Users are also free to modify the sample questions. Figure 3.24 shows the user interface after hitting the “Ask” button and receiving the answer as well as the other information as “evidence”.



**Answer(s)**

Lewis Hamilton  
Mercedes-Benz in Formula One

---

**Processed Information** +

**Chunked Sentence**

Who - won - German Grand Prix in 2016

---

**Answer Types Detected**

PERSON  
ORGANISATION

---

**Answer Type Classes**

<http://dbpedia.org/ontology/FormulaOneRacer>  
<http://dbpedia.org/ontology/FormulaOneTeam>

---

**Named Entities**

Label	URI	Score
German Grand Prix in 2016	<a href="http://dbpedia.org/resource/2016_German_Grand_Prix">http://dbpedia.org/resource/2016_German_Grand_Prix</a>	0.978723404255

---

**Mapped Phrases**

Phrase	Semantic Items
won	<a href="http://dbpedia.org/ontology/firstDriver">http://dbpedia.org/ontology/firstDriver</a> , <a href="http://dbpedia.org/ontology/firstDriverTeam">http://dbpedia.org/ontology/firstDriverTeam</a>

---

**SPARQL Queries**

```
SELECT ?x WHERE {<http://dbpedia.org/resource/2016\_German\_Grand\_Prix> <http://dbpedia.org/ontology/firstDriver> ?x .}
```

---

```
SELECT ?x WHERE {<http://dbpedia.org/resource/2016\_German\_Grand\_Prix> <http://dbpedia.org/ontology/firstDriverTeam> ?x .}
```

Figure 3.24: User interface of the QA system after hitting the Ask button

## Chapter 4

# Evaluation

In this chapter, first we provide statistical information about the ontology and the knowledge base in Section 4.1 and then in Section 4.2, we explain the experiments which are conducted to evaluate the accuracy of the knowledge base. Section 4.3 describes how we made use of the results in the evaluation of the QA system, performed by Blaauw, to study the potentials and limitations of the ontology and the knowledge base while being used in the QA system.

### 4.1 Statistical analysis of the knowledge base

In this section, we provide statistical information about the Knowledge base which reflects what types of knowledge is covered by the ontology and how much data is available in the KB. The KB contains 81004 RDF triples which describe schema and instance data of the ontology. Schema triples define the classes, sub-classes, and properties for the instances of classes as well as the restrictions for values of the properties (like “domain” and “range”). Instance data consists of all triples used for defining the instances of classes.

Regarding the instance data, there are 2153 instances in the KB which are distributed over six classes. Figure 4.1 shows this distribution as the number of instances per class in the ontology. According to this chart, the highest number of resources is defined for Grands Prix and drivers, whereas the lowest number of instances (resources) belongs to “Country” and “F1 Season” classes.

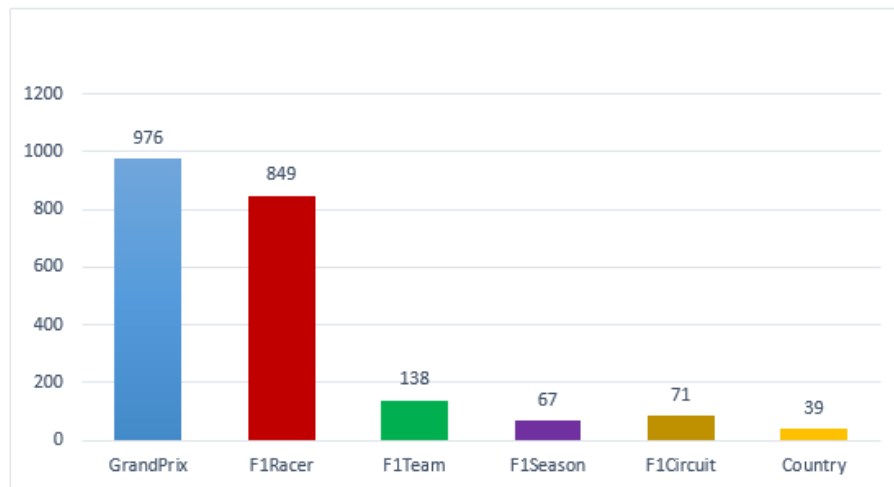


Figure 4.1: Number of instances in F1 ontology

The large number of resources for F1 races (Grands Prix) recorded in the KB reveals this fact that the knowledge base covers Formula One races over relatively a long period of time (67 years). In each year or season, a certain number of races has been recorded. Since 1950, on average, there have been 15 races hosted in each season. During the past 30 years, the average number of races increased and reached to a range between 15 and 20 races each season. By summing up these numbers, we end up with large number of instances recorded in the KB for Grands Prix (976 instances). The same logic applies for drivers. During the past 67 years of F1 races, vast number of athletes have participated in the races (849 drivers).

On the other hand, one could expect that the number of teams would also be in the similar range as “Grand Prix” and “F1 racer” instances, but teams do not have an “expiration date” as drivers have. They survive many years. For example, both Ferrari and Mercedes Benz have competed in F1 races since 1950’s.

Figure 4.2 shows how the number of instances defined for main classes in Formula One ontology (Grand Prix, F1 Racer, F1 Team) has changed over this 67-year period between 1950 and 2016. As mentioned before, during this time, we can see a steady rise in the number of Grands Prix, whereas for the number of drivers, the trend is almost the reversed. Since 1950 up to 1996, the number of drivers reduced gradually, and since then a considerable reduction in the number of attendants took place and remained with a little fluctuation until now (around 25 drivers each season). Highest and lowest numbers of participants (drivers) are recorded for seasons 1950 and 2008 respectively (100 versus 21).

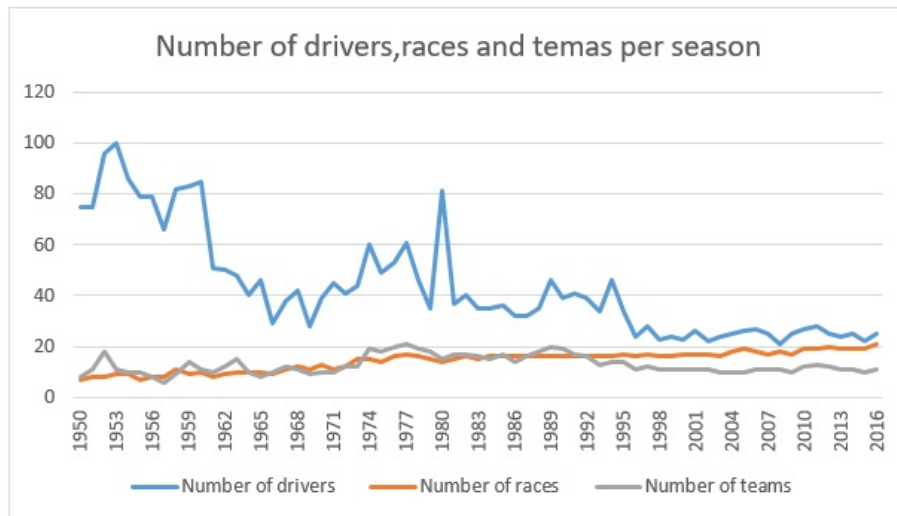


Figure 4.2: Number of instances per season

Moreover, in Figure 4.2, the graph of “teams” shows slight fluctuations in the number of participation ranging from 6 to 21 during this period. The highest number of different constructors that participated in the F1 races is recorded for season 1977 (21 teams). In contrast, the lowest number is recorded for season 1957 with only 6 entrants. Since 1998 until the present, about 11 teams on average have participated in the races.

Regarding the schema part of ontology, we determined the percentage of predicates (properties) that are specifically defined for each sample instance of F1 classes. The “type” property is common among all instances. The pie chart in Figure 4.3 shows this information for the main classes: “F1 Racer”, “F1 Team”, “F1 Season” and “Grand Prix”, where the most statistical information in domain of Formula One can be represented via the properties and values of resources that belong to these classes. As no specific property is defined for countries and F1 circuits in the ontology, they are not included in the pie chart. As we can see, the large numbers of predicates are defined for drivers and Grands Prix respectively, since main statistical information belong to these categories.

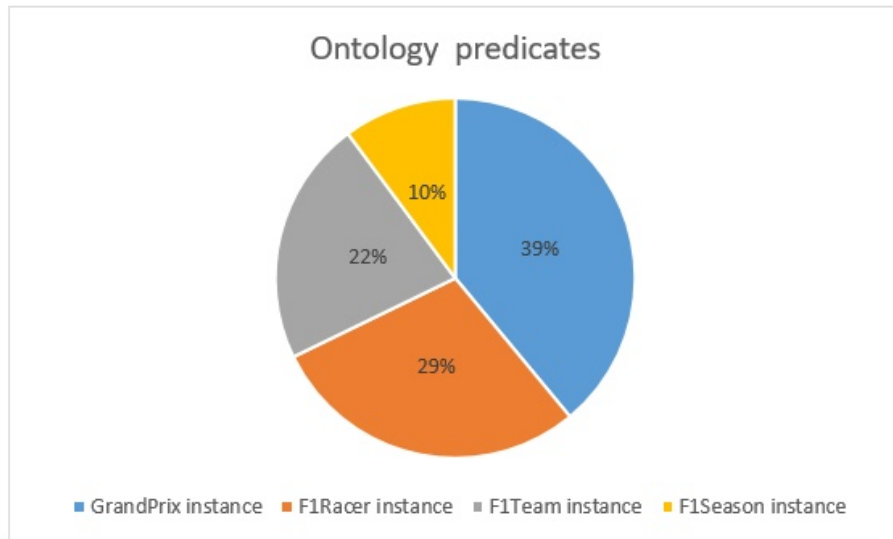


Figure 4.3: Percentage of defined predicates for instances of F1 classes

Finally, Figure 4.4 shows the dependencies between the six classes in the knowledge base. This figure is a snapshot that is taken from the Graph DB engine where all the ontology's knowledge is stored. The diagram shows the relationship between RDF classes, where a relationship is represented by links between the individual instances of two classes. Each link is an RDF statement where the subject is an instance of one class, the object is an instance of another class and the link is the predicate. Depending the number of links between the instances of two classes the bundle can be thinner or thicker and it gets the color of the class with more incoming links. The links can be in both directions.

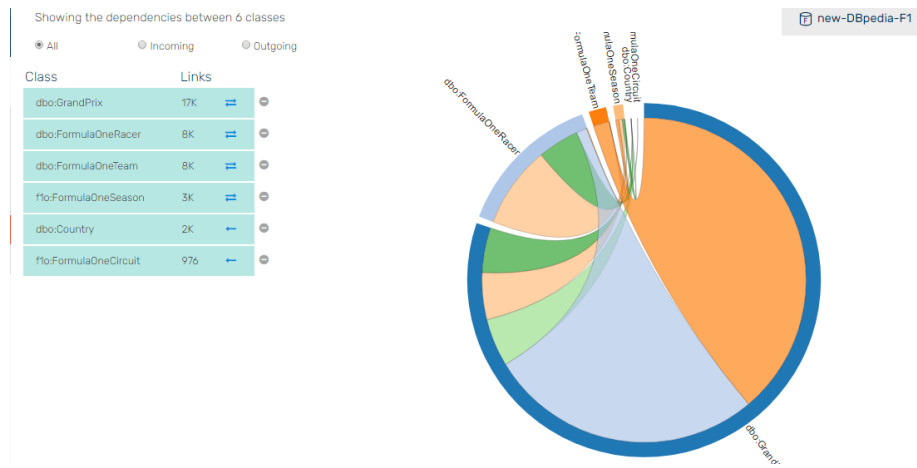


Figure 4.4: Relationships between Classes in F1 ontology

We can see in Figure 4.4, highest number of both incoming and outgoing links belongs to “Grand Prix” class, (by 16000 links), in contrast “F1Circuit” class has the lowest number of links with only 976 incoming links. By clicking on the “links” icon shown in the figure, we gained more details about the classes that are related to each F1 class. In summary, “Grand Prix” class has relationships with all the rest five classes and has most mutual relationships with drivers and teams. Moreover, Instances of “F1 Season” class have most relationship with the drivers.

## 4.2 Measuring the accuracy of the knowledge base

Evaluation of the knowledge base in terms of accuracy, has high importance and affect in accurate performance of the entire answering engine (QA system), since this knowledge base, first, is used as the main source for making the lexicons in “Question Processing” module and, second, as the main source of answers in “Answer Extraction” module of the QA system.

In this section, we plan to evaluate the accuracy of the knowledge base. In other words, we want to know how accurate our Knowledge Base (KB) is? Accuracy has two components, which correspond to the ideas of “precision” and “recall” in information retrieval.

### 4.2.1 Measurement of precision

To measure precision, we would like to know whether all the facts present in the created knowledge base (stored as triples) are correct against the source of origin. In our study, this source would be Wikipedia which is used as the main source of data in our KB.

In practice, since the number of available facts (triples) in our knowledge base is relatively large and obtaining the entire data is too time consuming, we decided to select a sample group of triples that can be a representative of all triples in the KB. To do sampling, we chose random sampling approach. The precision of the chosen triples, is measured against the corresponding facts available in Wikipedia pages. This precision can be later generalized to the precision of the whole knowledge base.

As mentioned in previous sections, the triples in our KB are classified in two main parts: KB schema and instance data. Before doing random sampling out of this collection of records, we decided to filter out the triples that are not the target of the evaluation, because either there is no explicit reference knowledge base that can be used for measuring correctness of them or their accuracy does not have a high importance in evaluating precision of the knowledge base. Therefore, the records stored in “schema” directory which include terminological knowledge of the ontology; not the statistical data about the instances, are ignored.

According to the explanation above, to specify target data, we selected the main directories in the knowledge base that include the instance data: “drivers”, “Grands Prix”, “teams” and “seasons” to choose samples from. “countries” and “circuits” directories contain triples that only define “type” of instances and do not have any predicates to cover the statistical data, thus we removed them from the target data.

To determine the size of sample, since the distribution of instance data is not symmetric over the main Formula One classes in the ontology, rather than choosing equal number of instances per class, we decided to choose more triples from “F1 Racer” and “Grand prix” classes which include larger number of instances and fewer triples from “F1 team” and “F1 season” classes which contain smaller number of instances. For this purpose, we chose one-tenth of instances from each class to make a relatively appropriate sample of 203 instances out of 2030 instances in main classes of the knowledge base. The sample size for each directory (class) as well as total number of instances in each class are presented in Table 4.1.

Number of instances					
Source	Grand Prix	F1 Racer	F1 Team	F1 Season	Total
Knowledge base	976	849	138	67	2030
Sample	97	85	14	7	203

Table 4.1: Number of instances in samples and the KB per class

To perform random sampling, by sending four SPARQL queries to the endpoint of the KB which benefit from “RAND” function for randomization, we created four sets of samples with the sizes determined in Table 4.1. Each set, consists of random instances related to each main class (directory). Then, using

SPARQL queries, we retrieved the related triples to all the instances in each set of samples. To specify the correct triples, we checked the correctness of triples in each set, against their associated fact recorded in Wikipedia pages (either in info-boxes or in tables of information about F1 entities). Finally, we used the formula below to measure precision values for each class.

“NCT” refers to the number of correct triples in sample set and “NTT” refers to the number of total triples in sample set.

$$\textit{Precision for each sample set} = \frac{NCT}{NTT} \quad (4.1)$$

Table 4.2 shows the results of this experiment:

<b>F1 Class</b>	<b>NTT</b>	<b>NCT</b>	<b>Precision value</b>
Grand Prix	2425	2403	0.99
F1 Racer	1275	1248	0.97
F1 team	154	154	1.00
F1 season	42	40	0.95
Total	3896	3845	0.98

Table 4.2: Detailed and total precision values of the samples from the KB

As shown in the table above, the precision value is high for each set of samples and totally we gained 98 percent precision for the 203 instances that were chosen out of 2013 instances in the KB.

### 4.2.2 Measurement of recall

To measure recall, we are interested to know if there are relevant triples in the ontology that are not present in the KB. In other words, we want to check what is supposed to be in the KB according to the ontology, and is not there.

Since there are lots of relevant information about all the instances recorded in our knowledge base, we decided to collect relevant information about only few instances from each main entity type as a set of samples. To select the instances, we did not follow a random approach, since we wanted to choose the properties or entities which might be challenging in terms of “recall”. For this purpose, we considered factor of “time” to choose various instances, thus we selected three resources for each entity type from three time periods that F1 races have been hold: “1950-1970”, “1970-1990” and “1990-2016”. The sample instances are listed in the table below (Table 4.3)



Sample instances from main F1 classes			
<b>Grand Prix</b>	1965 South African GP	1982 Swiss GP	2016 Canadian GP
<b>F1 Racer</b>	Alberto Ascari	Mario Andretti	Robert Kubica
<b>F1 Team</b>	Cooper	Shadow Racing Cars	Scuderia Ferrari
<b>F1Season</b>	1962 F1 Season	1978 F1 season	2015 F1 season

Table 4.3: Sample instances for recall evaluation

To determine the relevant information about a sample instance in a class, we used the predicates (properties) that are defined for instances of that class in the ontology. All these properties and their values are considered relevant information to the instance. Missing information for the sample resource, simply means lack of relevant predicates or their values for that resource in the KB.

According to the defined predicates for instances of “Grand Prix” class in the ontology, the direct information about “1965 South African Grand Prix” will be all the top three drivers and teams (winners, pole sitters, fastest ones) who participated in this Grand prix as well as the date, host country, circuit name and round number of this race in season 1965. Moreover, since “1965 South African GP” is part of “1965 F1 Season” entity, then all drivers and teams who attended this season can be considered the relevant information to the given instance. This part was provided in two lists from the Wikipedia pages to compare with the corresponding information in the knowledge base.

To check the existence of the first part (direct information), we simply looked them up manually in the knowledge base and compared them with the defined predicates for each instance of GP in the ontology, but for the second part, we tried to drive the corresponding information in two lists from our KB by sending SPARQL queries to the endpoint.

Finally, to check if there are any missing drivers or teams in the resulted lists, we made a comparison between the two sets of lists that we provided from Wikipedia and the KB. A similar investigation was performed for “1982 Swiss GP” and “2016 Canadian GP”.

For the driver “Alberto Ascari”, again according to ontology, the total records during his F1 career and personal information are considered relevant direct information. Moreover, all the seasons that this driver participated and all the teams which he had contract with during his F1 career, was considered as the relevant information.

For the detailed records, for simplicity, we decided to restrict the relevant information to one specific season. For example, in season 1952, all his records (number of races, wins, poles, podiums, fastest laps, podiums, position, and points) as well as the constructor with which entered the races on that season, are considered the relevant information about this driver. (Detailed records of the driver in each race of a season are not considered in our ontology, thus they

are not mentioned as the relevant information.)

Like the case of Grand Prix, we collected the mentioned information above by sending related queries to the KB, and made comparison with corresponding data in the Wikipedia table and information box for this driver to find the missing records (values). This approach was repeated for the other two instances of drivers.

The relevant information about “Cooper”, would be all the drivers who had contracts with this team and even became champion by this team, and the seasons that this team took part. This data is available in “Racing record” table on the Wikipedia page specific to this team. In addition, general statistics like number of races, points, poles, podiums, wins and fastest laps which this team gained during the history of F1 are considered relevant. (The detailed records of this team in each season or Grand Prix are not addressed in the ontology, thus they are not considered relevant.)

Like the previous approaches, most of the information mentioned above like “general F1 statistics” was accessible through direct queries. To specify drivers who became champion with this team, we used a query in the KB which looks up the winning drivers of Grands Prix whose first team is “Cooper”. This approach was repeated for the rest of instances.

At last, for “1962 F1 season” and the other two instances, according to the ontology, we considered the season date, first host country, number of races and season champions (driver and team) as the relevant information. In addition, all drivers and teams who competed in this season can be relevant data (which is accessible in Wiki-page of the sample season). Then like the previous experiments, we looked for the corresponding information in the KB.

Table 4.4 shows which relevant information found missing for each instance. “Missing information” in the table, refers to lack of predicate(s) or the values of predicates for an instance.

Sample instance	Missing information
1965 South African GP	8 participant teams in season 1965
1982 Swiss GP	1 participant team in season 1982
2016 Canadian GP	Fast team and pole team, distance laps, 2 teams and 4 drivers from season 2016
Alberto Ascari	Name and birth date, detailed records, (poles, points, etc.) in season 1950
Mario Andretti	Name and birth date, his team in 1982
Robert Kubica	Name and birth date, detailed records in season 2011
Cooper	10 drivers who drove with this team, 2 seasons
Shadow Racing-Cars	2 drivers who drove with this team, 2 seasons
Scuderia Ferrari	2 seasons this team participated
Season 1962	1 participant team
Season 1978	1 participant driver
Season 2015	—

Table 4.4: Missing data found for sample instances

The results above show that in all main directories, there are some relevant data that are missing. To provide recall information in terms of certain quantity, we defined some variables and calculated their values according to the results of experiments mentioned above. Table 4.5 shows results of these calculations. “NRR” refers to the number of relevant predicates for each instance that is retrieved from the KB. “NRK” refers to the number of relevant predicates for the instance in the ontology and “recall value” for each instance refers to fraction of relevant predicates for that instance in the ontology that are retrieved from the KB. Formula 5.1 shows how recall measure is calculated based on above definitions and variables.

$$\text{Recall for each instance} = \frac{NRR}{NRK} \quad (4.2)$$

Sample instance	NRR	NRK	Recall value
1965 South African GP	26	27	0.96
1982 Swiss GP	25	27	0.93
2016 Canadian GP	21	27	0.78
Alberto Ascari	14	17	0.82
Mario Andretti	14	17	0.82
Robert Kubica	13	17	0.76
Cooper	13	15	0.87
Shadow Racing-Cars	13	15	0.87
Scuderia Ferrari	14	15	0.93
Season 1962	6	7	0.86
Season 1978	6	7	0.86
Season 2015	7	7	1.0

Table 4.5: Recall values for sample instances

To calculate the overall recall value for the sample classes, we calculated the average of recall values of the samples in each class (recorded in Table 4.5). Table 4.6 shows the results of these calculations. According to the results, within the sample set from the knowledge base, the most amount of recall belongs to “Season” category whereas, the least amount of recall belongs to “Racer” category. (0.90 versus 0.80), and in total, recall rate for the chosen samples from the knowledge base is 87 percent.

F1 Class	Recall value
Grand Prix	0.89
F1 Racer	0.80
F1 team	0.89
F1 season	0.90
Total	0.87

Table 4.6: Detailed and total recall values for sample classes

### 4.3 Evaluating design of the ontology

In this part, we wanted to know how well the ontology is designed to help the QA system to answer the questions correctly. Two approaches helped us to perform this evaluation: preparing and using a set of base line questions intended to measure the accuracy of the QA system and an error analysis of the system after measuring the accuracy of the system.

As the QA system works based on the ontology, the preparation of base line questions has been performed as a shared task between Blaauw and me, but measuring the accuracy of the QA system and analysis of unsuccessful performance of some “Question Processing” components, have been covered in Blaauw’s thesis. In this study, we just use the results of the evaluations and analysis to evaluate the strengths and weaknesses of the ontology and the KB applied to the system. The following section explains how we prepared the baseline questions.

### 4.3.1 Baseline questions

We know that the main objective of the QA system is to provide a system in a friendly environment to interact with computers in natural language. The system does not require user to learn any programming language skills, any structure of a formal query (like SQL or SPARQL) or to know the structure of the knowledge base, but the user must know something about the Formula One domain to ask questions about the domain concepts like pole sitters, laps, podiums, winners, etc.

By considering the assumptions above, we prepared manually, 50 questions in “natural language” in English that are predicted to be asked from the system known as “baseline questions”. To prepare this set, we tried to avoid using keywords in the questions that corresponds to the domain-specific vocabularies. For example, instead of using “first driver” in a question to address the winner of Grand Prix, we used an equivalent term like “winner” which seems more natural to use in everyday language.

To determine the scope of questions and answers, we were inspired by the competency questions that were used for designing of the ontology, specifically targeting the domain knowledge of Formula One which is included in our ontology/ knowledge base. There are also few numbers of questions in F1 domain that cannot be answered by the ontology, just to challenge the system to handle the questions outside its domain.

Moreover, to challenge the system to answer various questions, we made the questions that look for different resources and properties in the ontology with different answer types. However, there are few numbers of questions in the set with similar meaning but different formatting in expression, just to test the system’s behavior to address these types of questions as well. For instance, the question *Who came in third place in 2014 Grand prix in Italy?* is equivalent to the question *Who was the third driver in 2014 Italian Grand Prix?*

Two kinds of classifications are considered for setting baseline questions. In the first one, the questions are classified as factoid and list. Definition of this classification as well as the number of questions in each category are provided in Table 4.7. This classification is based on the answer type: a single factoid answer versus multiple factoid answers.

Category of question	Description	Number of questions
Factoid	Questions with Single text answer	35
List	Questions with multiple text answers	15

Table 4.7: Categories of questions based on the answers

For each question in the baseline set, we manually created a SPARQL query that could retrieve the answer from the knowledge base. This could help us both to check the possibility of retrieving answers from the knowledge base and providing “ground truth” answers to compare them with the final answers of the QA system.

For the second classification, we considered the difficulty of questions and focused on these SPARQL queries. The difficulty of questions is determined based on the type and number of RDF statements which are used in making the corresponding SPARQL queries. The terms included in SPARQL queries are taken from the designed F1 ontology.

Type of question	Description	Number of questions
Type 1	At most two RDF statements in the SPARQL query	22
Type2	More than two RDF statements in the query (Filtering, ordering and setting limitations are allowed)	15
Type 3	Aggregated queries with grouping, summing, averaging or counting	10
Type4	SPARQL query is not enough to retrieve the answer (additional processing is needed)	4

Table 4.8: Types of questions based on difficulty

Based on this classification, in the set of baseline questions, we can consider the “Type 1” questions as the simplest ones and “Type 4” questions as the most difficult ones for the system to be processed.

Figure 4.5, represents a sample list of base line questions which shows the corresponding SPARQL query, type, category, and answer for each question.

Question ID	Question	Sample SPARQL Query	Type	Category	Answer
CQ1	Who won the Australian Grand Prix in 2011?	SELECT ?driver WHERE { dbr:2011_Australian_Grand_Prix dbo:firstDriver ? driver . }	1	Factoid	Sebastian Vettel
CQ21	Which drivers had the pole positions in the 2011 Season	SELECT ?drivers WHERE { ?x f1p:hasRecords ?node . ?node f1p:season f1r:2011_Formula_One_Season. ?node dbo:poles ?pole . FILTER(?pole > 0) }	2	List	Sebastian Vettel Mark Webber Lewis Hamilton
CQ24	What was the most points a driver got in the 2015 season?	SELECT ?point WHERE { ?driver f1p:hasRecords ?node . ?node f1p:season f1r:2015_Formula_One_Season. ?node f1p:points ?point . } ORDER BY DESC(?p) LIMIT 1	2	Factoid	381
CQ47	Which team has the most podiums in the history of F1?	SELECT ?team (MAX(?p) as ?podium) WHERE { ?team rdf:type dbo:FormulaOneTeam. ?team dbo:podiums ?p . FILTER(?p > 0) } GROUP BY (?team) order by desc(?podium) limit 1	3	Factoid	Scuderia Ferrari
CQ40	Which driver had the most consecutive wins since 2010?	-----	4	Factoid	Lewis Hamilton

Figure 4.5: A sample list of baseline questions

In this process, we found out that the ontology has the potential to answer all the defined questions in the set except for “Type 4” questions, but this cannot be considered a big advantage for the ontology design, because we knew beforehand which questions could be answered by the ontology and included them in the set. Stronger evaluation occurs when the ontology is used by the machine (QA system) to create SPARQL queries. The complex SPARQL queries in the set of questions, indicate the probable limitations in the design of ontology, for example there was no simpler relation between two classes or there was not a property to provide value in an easier way by the query.

The next section, evaluates the design of ontology based on the results of performance evaluation of the QA system and analysis of errors in answering the base line questions.

### 4.3.2 Analysis of errors

Referring to Blaauw’s report, measuring the accuracy of the QA system was performed in two levels: end-to-end and component levels. In end-to-end evaluation, the system was fed by the baseline questions and the final results were compared to the ground truth. The results showed that the QA system answered 22 out of 50 questions correctly and yields an accuracy score of 40% and highest score of accuracy belongs to “Type 1” questions which were considered easy questions. In component evaluation, performance of each component in pipeline of processing questions, were evaluated, then the probable causes of weak performance of some components were recognized. More details about these evaluations are provided in Blaauw’s work.

During the error analysis, as the focus of this study is on development of the ontology and the KB, we focused on the problems that were mostly related to design of the ontology or lack of data in the KB. Eight questions out of 28 problematic questions were recognized in the baseline set, that are related to these error types. These questions are listed as below:

- Q11. *List the teams that participated in the 2015 season?*
- Q25. *Name a circuit in Spain?*
- Q35. *List all the drivers who has won the Australian Grand Prix more than once?*
- Q37. *List all the teams who have won the Italian Grand Prix more than once?*
- Q41. *Who is the youngest driver to become a champion?*
- Q42. *Who is the youngest driver with more than one win?*
- Q43. *How many points did Nico Rosberg get in the Canadian Grand Prix in 2014?*

The list below, explains the probable lacks or limitations in the ontology, related to each question listed above:

- Regarding “Q11”, in schema part of the ontology, the relationship between individual instances of “Team” class and “Season” class, are not mutual, instead, they are mostly incoming links which implicitly means that looking up information for team instances relevant to seasons, should start from seasons (the origin of links) not the teams, thus making the search, difficult for the system.
- Regarding “Q25”, in schema part of the ontology, there is no direct relation between a F1 circuit and the country it is located in. The system first needs to look up the Grand Prix, the hosting country, and the circuit where the race took place.
- Regarding “Q35” and “Q37”, as mentioned before, “Australian GP” or “Italian GP” is not defined a separate entity in the ontology. All instances of “Grand Prix” Class have date information in their definition like “2014 Australian GP” which is an instance of “Grand Prix” class, thus it might cause problem in concept mapping.
- For “Q41” and “Q42”, there is no data about birth date of drivers in the KB and hence, the system cannot use this attribute.
- For “Q43”, there is no defined relation/property between instances of drivers and Grands Prix that allows storing records of each driver in a specific Grand Prix, the KB just contains the detailed records of drivers in each F1 season.



In summary, the recognized lacks in design of ontology can be divided in three parts: lack of relevant entities in domain of Formula One (like Grand Prix entity with name of the host country), lack of specific properties for some entities (like “has Records in GP” for drivers), lack of enough mutual relationships (two-way links) between some entities (like relation between “Team” and “Season” or between “Circuit” and “Country”). Finally, the lack of data in the KB is related to lack of values for “birth Date” property of “Racer” instances.

## Chapter 5

# Conclusion

This chapter consists of four sections. First section, provides a brief overview of what we covered in this report. Section 5.2 Discusses our results and findings from the experiments. The limitations of the work are explained in Section 5.3. Finally, our suggestions for future work are presented in Section 5.4

### 5.1 Achievements

In this study, we managed to develop a conceptual model (ontology) in domain of Formula One and make a knowledge base (KB) for a QA system, using semantic technologies and reliable information sources.

Through the design process of ontology, we have shown that how to provide specifications of target ontology using competency questions and Wikipedia as source of answers to the questions. During conceptualization of ontology, we have shown how to structure the domain knowledge in a conceptual model, using information from DBpedia ontology; where information from Wikipedia pages and tables are extracted and stored in a structured format using semantic technologies like XML and RDF.

Furthermore, we have demonstrated how to extend the DBpedia ontology in domain of formula One to a more advanced ontology with larger number of entities, properties (attributes) and relations between entities to be able to answer variety of statistical questions, mentioned in the set of competency questions.

Through the implementation process of ontology, we could collect all the required data specified in the extended conceptual model in two parts: data which represents terminological knowledge of ontology and data which defines instances of entities in the target domain. We also managed to formulate all the gathered data in RDF statements with N-triples format which are understandable for machines/ QA system to process.

In the final step of implementation, we managed to store data in a graph knowledge base which uses graph structures for semantic queries to represent and store data. The graph structure was advantageous to relate and link data

items together directly which mostly makes retrieval of data with one operation through a SPARQL query.

Finally, to generate a real version of the QA system, we implemented a prototype for the system. This prototype makes use of all components of the QA system which were designed for processing the questions and generation of answers as well as a graphical user interface which provides online interaction with the user. The prototype was implemented through a web application that uses Flask framework to run Python codes inside HTML tags.

The results and findings from evaluation of our ontology and knowledge base are discussed in the next section.

## 5.2 Discussion

The statistical analysis of our knowledge base shows that we could store all instances of drivers, teams, and Grands Prix during 67 years of F1 races along with many relevant information about them. Experiments that we conducted for measuring the accuracy of the knowledge base on a subset of records, showed that at least we have 90 and 80 percent precision and recall rates, respectively, for the chosen samples and if we can generalize this to the whole KB, it means we can rely heavily on the accuracy of the knowledge base as the source of answers in our QA system; the KB which reflects both schema and instance knowledge in our ontology.

Furthermore, referring to Blaauw's report, results of experiments on measuring the accuracy of the QA system indicate that the system works well to answer easy and factoid questions about statistics of Formula One whose answers are simple facts, which are directly available in the KB and the corresponding automated SPARQL queries, do not need more than two statements and applying advanced aggregation functions.

Evaluation of design of the ontology, through performing component evaluation and analyzing errors for unsuccessful results, reveals that there are some lacks in schema part of the ontology like lack of a relevant entity, a property or mutual relationship between some entities (like between "Team" and "Season" or between "Team" and "Driver" entities); which makes interpreting the questions, difficult for the system while processing the questions.

However, we should mention that effectiveness and efficiency of the designed ontology is dependent on the system or application that makes use of the ontology. It implicitly means, if we used this ontology in another QA system which leveraged from other techniques for interpretation of NL questions; our evaluation about effectiveness of the ontology was most probably different from the current assessment.

From the results of our experiments, we have learned that, design of a complete and effective ontology in a specific domain is a heavy task that need enough time and expertise to deal with different issues. For example, selection of appropriate set of competency question which cover the scope of most potential questions plays important role in defining all necessary entities and relationships

between them in the conceptual model. We also learned that, reusing existing ontologies, like DBpedia ontology, can save time and effort in making an ontology from the scratch, thus leading us to concentrate more on removing existing lacks in current ontology for developing a more complete ontology.

Moreover, after evaluation of the QA system, we found that choosing meaningful and distinguishable names for entities, properties, and relations, makes ontology more readable and easier to use for the QA system. Since, during interpretation of questions, detecting entities in the questions and generation of formal queries are based on the entities and relations defined in the ontology.

In addition, as we could answer 90 percent of baseline questions, based on the ontology, through manual generation of SPARQL queries; we can conclude that the ontology has required potential to answer many statistical questions that it was designed for, but capabilities of an ontology in providing clear knowledge are mainly determined when it is used as a conceptual model in a real application or system. The results of using our ontology in the QA system proved that we cannot expect the current QA systems can totally compete with human mind in reasoning and comprehension, therefore we learned to consider the strengths and weaknesses of computer processing while developing an ontology.

### 5.3 Limitations

In design of ontology, although we tried to make various competency questions in the domain of interest, but still, there are many more questions that potential users can ask about statistics of Formula One that we did not cover in the set of competency questions. As a result, definition of all elements in ontology was restricted to the limited and certain number of concepts and entities mentioned in this set of questions.

Moreover, we did not have time to collect more relevant data about some entities like “Driver”, “Team” and “Season”. For example, our KB does not include information about detailed records of each team in each season or Grand Prix; or it does not cover information about detailed records of drivers in each Grand Prix during a season like “poles”, “points”, “podiums”, etc.

The ontology language that we chose for writing schema part of ontology (RDFS) is not that expressive and cannot describe resources in sufficient details.

We also did not have time to develop automatic approach to be able to measure the accuracy of all records in the KB, therefore restricted ourselves to a small sample of KB in evaluation process.

The baseline questions that we set manually for both end-to-end and component evaluations, like the case of competency questions, could not be a comprehensive representative of all questions that might be asked from the system in domain of Formula One.

## 5.4 Future work

In this section, we present the following suggestions for future work. Regarding optimization of ontology, further work includes new definitions for missing entities, properties, and mutual relationships. Moreover, adding more relevant data to the defined resources, can convert the knowledge base to a more comprehensive source for the QA system to answer a wider range of user's question in domain of Formula One. Extending the knowledge of ontology to the knowledge in domain of other sports, also can be helpful for the future QA systems within sports domain that need ontologies or semantic data sources.

Regarding the preparation of answers, improving the performance of the related component by adding NLP capability, would make it to produce the result in a more natural way and would give a sense of real interaction with the system to the user. For example, if the answer of a question that looks for a winner of "2012 Australian GP" is shown as *Jenson Button* as a link-able text, it would be more interesting if the answer would be processed like this: *The winner of Australian GP in 2012 is Jenson Button.*

Finally, about the GUI, we can add more flexibility to the system by adding an "edit" part to the user interface which allows the users with technical knowledge, to cooperate with generation of desired answers by editing the SPARQL queries made by the system.



# Appendices

## Appendices as attachments

All the main documents that were created during design of the ontology in two phases such as competency questions; the baseline questions as well as the corresponding SPARQL queries and answers; are collected in three folders and have been attached to this report as a zipped file. The attached documents are introduced in summary, in the table below:

<b>Table of contents</b>	
<b>Content of folder</b>	<b>Label of folder</b>
Ontology documents -Specification	A
Ontology documents -Conceptualization	B
Baseline Questions	C

Table 1: Contents of attachments





# Bibliography

- [1] S. A. Aroussi, N. E. Habib, and O. E. Beqqali. “Improving question answering systems by using the explicit semantic analysis method”. In: *2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA)*. 2016, pp. 1–6.
- [2] Abdi Asad, Idris Norisma, and Ahmad Zahrah. “QAPD: an ontology-based question answering system in the physics domain”. In: *Soft Computing* (2016), pp. 1–18. ISSN: 1433-7479. URL: <http://dx.doi.org/10.1007/s00500-016-2328-2>.
- [3] Asma Ben Abacha and Pierre Zweigenbaum. “MEANS: A medical question-answering system combining NLP techniques and semantic Web technologies”. In: *Information Processing Management* 51.5 (2015), pp. 570–594. ISSN: 0306-4573. URL: <http://www.sciencedirect.com/science/article/pii/S0306457315000515>.
- [4] T. Berners-Lee and M. Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. Harper San Francisco, 1999.
- [5] Christian Bizer, Tom Heath, and Tim Berners-Lee. “Linked data-the story so far”. In: *Semantic services, interoperability and web applications: emerging concepts* (2009), pp. 205–227.
- [6] Christian Bizer et al. “Linked Data on the Web (LDOW2008)”. In: *International Journal on Semantic Web and Information Systems (IJSWIS)* (2008).
- [7] Abdelghani Bouziane et al. “Question Answering Systems: Survey and Trends”. In: *Procedia Computer Science* (2015), pp. 366–375.
- [8] Martin Dzbor et al. “Neon-life cycle support for networked ontologies (overview and objectives)”. In: *2nd European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology* (2005).
- [9] *Experimental Factor Ontology*. Web Page. 2015. URL: <https://bioportal.bioontology.org/ontologies/EF0>.
- [10] Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. “Methontology: from ontological art towards ontological engineering”. In: *AAAI Technical Report SS-97-06* (1997).

- [11] Macgregor George. "Introduction to a Special Issue on Digital Libraries and the Semantic Web: Context, Applications and Research." In: *Library Review* 57 (2008), pp. 173–177.
- [12] Asunción Gómez-Pérez, Mariano Fernández, and A de Vicente. "Towards a method to conceptualize domain ontologies". In: *van der Vet P (ed) ECAI's 96 Workshop on Ontological Engineering. Budapest, Hungary* (1996).
- [13] *Graph database*. 2016. URL: <https://en.wikipedia.org/wiki/Graph%20database>.
- [14] Bert F Green Jr et al. "Baseball: an automatic question-answerer". In: *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*. ACM, pp. 219–224.
- [15] Thomas R Gruber. "Toward principles for the design of ontologies used for knowledge sharing." In: *International Workshop on Formal Ontology* (1993), pp. 907–928.
- [16] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. third edition. 2016.
- [17] Kamath et al. "A semantic search engine for answering domain specific user queries". In: *Communications and Signal Processing (ICCSP), 2013 International Conference* (2013).
- [18] Jens Lehmann et al. "DBpedia—a large-scale, multilingual knowledge base extracted from Wikipedia". In: *Semantic Web 6.2* (2015), pp. 167–195.
- [19] Sweta P Lende and MM Raghuvanshi. "Question answering system on education acts using NLP techniques". In: *Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave), World Conference on*. IEEE, 2016, pp. 1–6.
- [20] Andrea Marchetti et al. "Formalizing Knowledge by Ontologies: OWL and KIF". In: *CNR, IIT Department, Via Moruzzi 1 I-56124* (2015).
- [21] Nelson SJ. McCray AT. "The representation of meaning in the UMLS". In: *Methods of information in medicine* 34 (1995), pp. 193–201.
- [22] Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. 2001. URL: <https://protegewiki.stanford.edu/wiki/Ontology101>.
- [23] *Ontology Components*. 2009. URL: [https://en.wikipedia.org/wiki/Ontology\\_components](https://en.wikipedia.org/wiki/Ontology_components).
- [24] Thomas B. Passin. In: *Explorers' guide to the Semantic Web*. Manning Publications, 2004. Chap. one.
- [25] Pascal Hitzler Rudolph., Markus Krötzsch, and Sebastian. *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC, 2009.
- [26] M. Ruslan and Navigli Roberto. 2016. URL: <http://www.oxfordhandbooks.com/>.

- [27] Kim André Sand. “Question Answering using Syntactic Patterns in a Contextual Search Engine”. Thesis. 2006.
- [28] Awny Sayed and Amal Al Muqrishi. “IBRI-CASANTO: Ontology-based semantic search engine”. In: *Egyptian Informatics Journal* (2016). ISSN: 1110-8665. URL: <http://www.sciencedirect.com/science/article/pii/S111086651730004X>.
- [29] Schmachtenberg et al. Figure. 2014. URL: <http://lod-cloud.net/>.
- [30] Hellmann Sebastian. “Community-Driven Engineering of the DBpedia Infobox Ontology and DBpedia Live Extraction”. Thesis. 2010.
- [31] *Semantic Search*. 2016. URL: [https://en.wikipedia.org/wiki/Semantic\\_search](https://en.wikipedia.org/wiki/Semantic_search).
- [32] *Semantic Web*. 2011. URL: [https://en.wikipedia.org/wiki/Semantic\\_Web\\_Stack](https://en.wikipedia.org/wiki/Semantic_Web_Stack).
- [33] Tirpude Shubhangi and A.S. Alvi. “Closed Domain Question Answering System: A Survey”. In: *International Journal of Informative and Futuristic Research* (2015), pp. 3347–3353.
- [34] *Sparql Query Language for RDF*. 2013. URL: <https://www.w3.org/TR/rdf-sparql-query/>.
- [35] *Triple store*. 2016. URL: <https://en.wikipedia.org/wiki/Triplestore>.
- [36] Mike Uschold and Michael Gruninger. “Ontologies: Principles, methods and applications”. In: *Knowledge Engineering Review* 11 (1996).
- [37] *W3C Semantic Web Activity*. 2001. URL: <https://www.w3.org/2001/sw/>.
- [38] Vivienne Waller. “Making knowledge machine-processable: some implications of general semantic search”. In: *Behaviour Information Technology* 35.10 (2016), pp. 784–795. ISSN: 0144-929X. URL: <http://dx.doi.org/10.1080/0144929X.2016.1183710>.
- [39] Ron. Weber. “Ontological Foundations of Information Systems”. In: *Coopers and Lybrand Accounting Research Methodology Monograph* (1997).