



SEMANTIC WEB

Can the Semantic Web be combined with AI? For example, Natural Language Processing to create a smart search engine?

DETAILS

A dissertation submitted in partial fulfilment of the requirements for the degree of Bachelor of Science (Honours) in Software Engineering.

By Ieuan Walker

Department of Computing & Information Systems
Cardiff School of Management

Cardiff Metropolitan University

April 2017

Declaration

I hereby declare that this dissertation entitled '*Semantic Web* – Can Semantic Web be combined with AI?' is entirely my own work, and it has never been submitted nor is it currently being submitted for any other degree.

Candidate: Ieuan Walker

Signature:

Date:

Supervisor: Dr Ambikesh Jayal

Signature:

Date:

Abstract

The semantic web is the next evolution of the world wide web. There are many names for the semantic web such as linked data, the web of data and even web 3.0.

What is the semantic web? Well to summarise my dissertation, the semantic web is still in development but it is a set of new languages and technologies that are used to connect data across the web. But it doesn't only connect different resources across the web it also adds meaning and understanding to the data. With the combination of connected data and the meaning/ understanding of the data, it is enabling machines to process a lot of information on the web so that they can perform complex tasks for us.

Companies such as Apple, Google, and IBM are using the semantic web to enhance their AI systems. In my project, I will be utilising a small part of AI known as Natural Language Processing (NLP).

For my dissertation, I will build a semantic web search engine combined with natural language processing to build a smart search engine. You can find the entire project in the submitted documents or at the following link -

<https://github.com/IeuanWalker/Dissertation-Project>. I have also created a video demonstration of the web application which is also available in the submitted documents or at the following link - <https://youtu.be/IuJhd5diDgg>.

Acknowledgements

I would like to thank my dissertation advisor, Ambikesh Jayal, for his patience, motivation, enthusiasm, throughout my project.

Getting through my dissertation required more than academic support. I would like to thank my family and friends, for listening to and, at times, having to tolerate me over the last three years.

Table of Contents

Declaration.....	i
Abstract.....	ii
Acknowledgements.....	iii
List of Tables	vi
1. Introduction	1
1.1 Why modern day search engines are failing.....	2
1.2 Possible uses for semantic web	3
2. Aims and Objectives.....	4
3. Literature Review.....	5
3.1 Linked Data, how does it differ from the Semantic web?	6
3.2 Data Interchange - Resource Description Framework (RDF)	7
3.3 Querying language - SPARQL.....	8
3.4 What is the difference between RDFS and OWL	11
3.5 Taxonomies – Resource Description Framework Schema (RDFS)	12
3.6 Ontology – OWL	13
3.7 Natural Language Processing (NLP)	14
4. Methodology.....	15
4.1 Quantitative research.....	15
4.2 Design methodology	16
4.2.1 Programming Language.....	16
4.2.2 Databases	17
4.2.3 Code Management	17
4.2.4 Natural Language Processor	20
4.2.5 Conclusion	21
5. Project Planning	22
5.1 Research Period.....	22
5.2 Development.....	22
5.3 Discussion and findings	23
6. Requirements and analysis	24
6.1 Languages and technologies	24

6.2	Natural Language Processing	24
6.3	DBpedia	25
7.	Design.....	26
7.1	Design Patterns	26
7.2	Database.....	26
7.3	Search process.....	27
7.3.1	Optimisation based on time of search to reduce calls across the network.....	28
8.	Implementation and testing	29
8.1	Designing the user interface	29
8.2	LUIS.....	30
8.2.1	Build you app	30
8.2.2	Improve and tweak.....	32
8.2.3	Test your app	32
8.2.4	Publish your app	32
8.3	Creating SPARQL query	33
8.4	Generating a SPARQL Query	33
8.5	SPARQL search.....	39
8.6	Database – Entity framework code first approach	39
8.6.1	Deleting from the database.....	40
8.6.2	Adding to the database	41
8.7	Search Process.....	42
9.	Results and discussion	44
9.1	Limitations.....	44
9.2	Current search engines VS a semantic search engine	45
9.3	The combination of different technologies	45
9.4	Conclusion	47
	References	48
	Appendix	52
	Appendix 1 – Ethics Approval Form	52

List of Tables

Table 1

Search String	Update status	Time (ms)
10 soundtrack films from 2001	Need updating	1377
10 soundtrack films from 2001	Need updating	2964
10 soundtrack films from 2001	Need updating	1339
Average		1893

Table 2

Search String	Update status	Time (ms)
10 soundtrack films from 2001	Already updated	57
10 soundtrack films from 2001	Already updated	61
10 soundtrack films from 2001	Already updated	65
Average		61

Table 3

Search String	Update status	Time (ms)
World music films	Need updating	3299
World music films	Need updating	1719
World music films	Need updating	3654
Average		2891

Table 4

Search String	Update status	Time (ms)
World music films	Already updated	86
World music films	Already updated	126
World music films	Already updated	70
Average		94

Table 5

Search String	Update status	Time (ms)
Films from 2010	Need updating	7576
Films from 2010	Need updating	7732
Films from 2010	Need updating	8124
Average		7811

Table 6

Search String	Update status	Time (ms)
Films from 2010	Already updated	87
Films from 2010	Already updated	104
Films from 2010	Already updated	100
Average		97

1. Introduction

The web is constantly evolving. It is a growing universe of interlinked web pages and web apps. What the average user doesn't see are the languages and technologies that make it all possible.

Over time web pages went from static pages to full-featured web applications. And the technology behind it all has evolved immensely over time giving web developers the ability to create web applications that people didn't even think was possible at the beginning.

But what's next? Is it the semantic web?

Tim Berners-Lee, the inventor of the World Wide Web is leading the development of the next generation of web architecture- called the semantic web. The semantic web is described in an article from Scientific America *"The Semantic web is not a separate web but an extension of the current one, in which information is given web defined meaning, better enabling computers and people to work in cooperation."* (Berners-Lee, et al., 2001).

So why is it called semantic web? Well *"The word semantic itself implies meaning or understanding. As such, the fundamental difference between Semantic Web technologies and other technologies related to data (such as relational databases or the World Wide Web itself) is that the Semantic Web is concerned with the meaning and not the structure of data"* (Cambridge Semantics, n.d.).

There are a lot of names for the semantic web such as linked data, the web of data and even web 3.0. And you must remember that the semantic web is the next major evolution in connecting data across the web. The objective of the semantic web is to create a standard representation of linked data to allow machine processing on a global scale. The World Wide Web Consortium (W3C) is developing the open standard markup languages that would unleash the power of the semantic web.

So, the semantic web is about adding meaning to data on the web, so that web pages can be easily processed by machines. One of the core technologies of the semantic web is Universal Resource Identifiers (URI). Here is a similar reference from a journal written by Diana Maynard et al (Maynard, et al., 2016). A resource can be entities such as 'Philippe Coutinho', a concept such as 'Footballer' or a relationship describing how different entities relate to each other such as 'spouse-of'. I will go into a lot more detail of the technologies involved in the literature review section (*see section 3*).

Over the last few years there have been major breakthroughs in the world of Artificial Intelligence (AI), so in my dissertation, I will be combining the semantic web with AI. More specifically I will be touching on a small area of AI which is Natural Language Processing (NLP). I have decided to combine these two (semantic web and NLP) because they are both semantic technologies. But there is a major difference between them. The semantic web is

the process of adding meaning and understanding to structured data and NLP is the process of understanding unstructured data.

NLP *“is the automatic processing of text written in natural (human) languages (English, French, Chinese, etc.), as opposed to artificial languages such as programming languages, to try to “understand” it.”* (Maynard, et al., 2016). NLP can also be known as Natural Language Engineering (NLE) or Computational Linguistics (CL).

“Natural language input can permit a user to easily interact with an electronic device using well-known language” (Futrell & Gruber, 2016). This is what I intend my project to do. I will take a sentence from a user and use NLP techniques to determine the intent of the user and then use the semantic web to gather the relevant data.

The topic I have chosen for my dissertation is the semantic web. More importantly, I will be concentrating my dissertation on how to implement a semantic web search engine and include natural language processing to create a smart search engine.

1.1 Why modern day search engines are failing

In this modern day, most people have one tool to find relevant resources on the web, which is some sort of search engine, whether it's Google, Bing or something else. The main reason search engines were created was so users didn't have to remember URL for every site they wanted to visit.

“The world wide web is a big place. If you know the web address, or URL, of a site you can find it by typing it into the address bar along the top of your browser. But what if you don't know the URL?”

You can find pages by following links from other pages but usually, it is easier to search for things using a search engine.” (BBC, n.d.).

Search engines such as Google indexes the world-wide web by utilising Bots that crawl the web and index its findings. Each time a web page is changed or added a bot will soon index the page again. So, a search engine consists of three different sections which are bots that crawl the web, an indexing mechanism and a query interface. The system works by allowing users to enter keywords for the search then the system attempts to match the keywords in the indexes and return the most popular results.

So, the problem arises when the results are returned. As you may have experienced yourself, many of the returned results are irrelevant and off topic. This is because your keywords can be used in many different topic areas and there could be several meanings for the keywords (i.e. different languages).

Another reason a search result can be off topic is because of the lack of understanding of the context of the words and relationships between them. For example, if we query a search engine with a question like this 'Who is the head of America?', we could get back any

number of different results (see figure 1.1.1) but if we asked the same question to a person they might assume we are talking about the president and we'll get the result we are after.

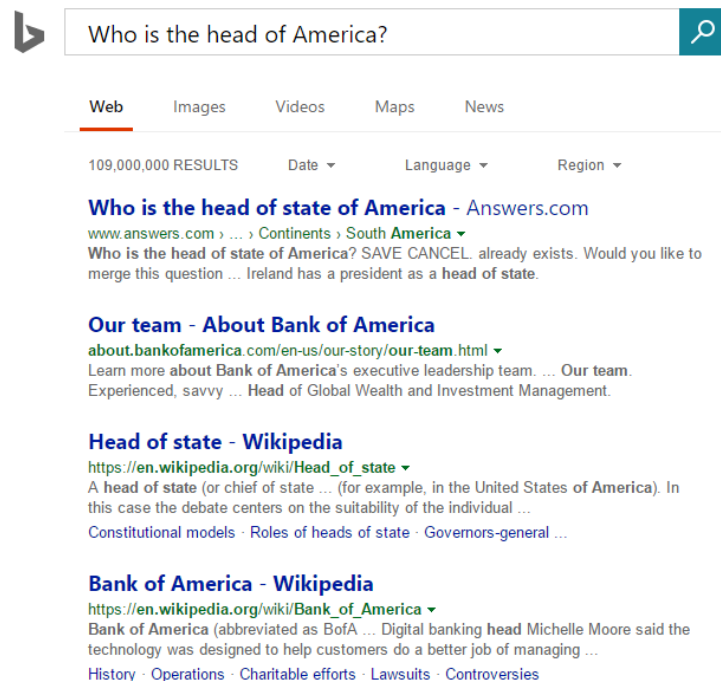


Figure 1.1.1 Example search on current search engines

If a search engine had the ability to understand the context/ meaning of the words within the sentence, then the returned result will be much more accurate. And this is just one of the goals of the semantic web.

1.2 Possible uses for semantic web

A search engine is only one of many different applications for the semantic web. There are a lot of possibilities for semantic web one of which Tim Berners-Lee explains in a ted talk.

"if you're looking at Alzheimer's, for example, drug discovery — there is a whole lot of linked data which is just coming out because scientists in that field realize this is a great way of getting out of those silos, because they had their genomics data in one database in one building, and they had their protein data in another. Now, they are sticking it onto — linked data — and now they can ask the sort of question, that you probably wouldn't ask, I wouldn't ask — they would. What proteins are involved in signal transduction and also related to pyramidal neurons? Well, you take that mouthful and you put it into Google. Of course, there's no page on the web which has answered that question because nobody has asked that question before. You get 223,000 hits — no results you can use. You ask the linked data — which they've now put together — 32 hits, each of which is a protein which has those properties and you can look at. The power of being able to ask those questions, as a scientist — questions which actually bridge across different disciplines — is really a complete sea change. It's very very important. Scientists are totally stymied at the moment

— *the power of the data that other scientists have collected is locked up and we need to get it unlocked so we can tackle those huge problems.*” (Berners-Lee, 2009)

As Tim Berners-Lee explains semantic web can help speed up research by allowing people to ask questions they couldn't ask before without spending a lot of time to answer the question themselves. But he goes on to say that semantic web will impact everyone's life as the semantic web is all about data. And virtually everyone has a little bit of data on the web whether it's on a social network or a shopping site. I'd recommend you watch Tim Berners-Lee Ted talk on the semantic web.

2. Aims and Objectives

My Dissertation topic is the semantic web and this will include a range of objectives and aims. My chosen research question is *“Can semantic web be combined with AI, for example, Natural Language Processing to create a smart search engine?”*

To enhance my research, I must answer several sub-questions before I can attempt to answer my research question. Here are a couple of these questions: -

- Why would a semantic search engine be better than a traditional search engine (Google/ Bing)?
- Can semantic web data be combined with a traditional relational database?

So, my first objective is to *“Investigate current search engines and explain how a semantic search engine will differ”*. In this objective, I will attempt to describe how a traditional search engine such as Google works and explain how a semantic search engine will be different.

Another objective is to *“Explain the benefits of semantic data”*. In this, I am going to research benefits of semantically linked data.

I will also *“Investigate if semantic data can be combined with a traditional database”*. To answer this question, I am going to have to create an SQL database.

The final objective is to *“Develop a simple semantic web application using natural language processing”*. In this objective, I plan to develop a semantic web search engine using several different languages and technologies to achieve this.

After answering the above questions and completing these objectives I hope to do a critical analysis of my findings and apply these findings to answer the main question.

3. Literature Review

In this section, I am going to discuss the current research and technologies on several related topics. The image below (*see figure 3.1*) shows all the different languages and technologies that have been created and utilised for the semantic web. These technologies and languages are recommended by the World Wide Web Consortium (W3C) and are currently the standards for the semantic web.

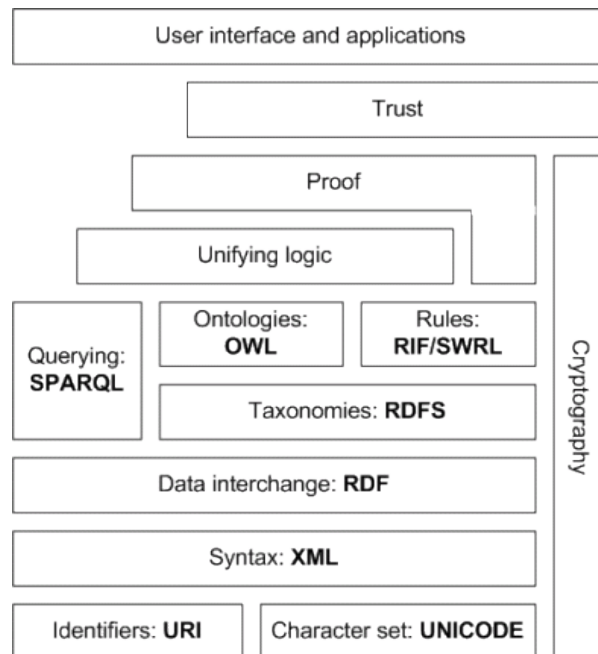


Figure 3.1 Semantic web architecture in layers – 'Layer Cake'

There has been a lot of research written about the semantic web and websites dedicated to learning more about it. Such as the article written by Tim Berners-Lee (the lead developer for the semantic web) called 'Linked Data – Design Issues' (Berners-Lee, 2006). Within the article, he states four rules for publishing data on the web, so that the published data becomes part of the single global data space:

1. "Use URIs as names for things"
2. "Use HTTP URIs so that people can look up those names"
3. "When some looks up the URI, provide useful information, using the standards (RDF*, SPARQL)"
4. "Include links to other URIs, so that they can discover more things"

As you can see URI, RDF and SPARQL are the core technologies and languages used for the semantic web and Cambridge Semantics cooperates this by stating *"The Semantic Web standards—RDF, SPARQL, OWL, and others—were merely drafts in 2001, but they have now been formalized and ratified."* (Cambridge Semantics, n.d.).

In the following section, I will be discussing data interchange (RDF), the querying language (SPARQL), taxonomies (RDFS) ontologies (OWL), and Natural Language Processing (NLP), to gain a better technical understanding of how the semantic web works.

3.1 Linked Data, how does it differ from the Semantic web?

When you first start to learn and understand what the semantic web is, you'll hear the term linked data. But what does linked data mean and how does it differ from the semantic web?

The semantic web is the idea of creating “a web of data” (w3c, n.d.). And linked data is the way to achieve the idea. So, linked data is the implementation of the semantic web.

In the book ‘Linked Data – Evolving the Web into a Global Data Space’ there is a quote that explains linked data, “*Linked Data provides a publishing paradigm in which not only documents, but also data, can be a first class citizen of the Web, thereby enabling the extension of the Web with a global data space based on open standards - the Web of Data.*” (Heath & Bizer, 2011).

And from an article written by Tim Berners-Lee (Berners-Lee, et al., n.d.) states “*while the Semantic Web, or Web of Data, is the goal or the end result of this process, Linked Data provides the means to reach that goal*”.

So, linked data is “a powerful means of structuring and publishing data on the web” (Meehan, 2014) to achieve the goals of the semantic web.

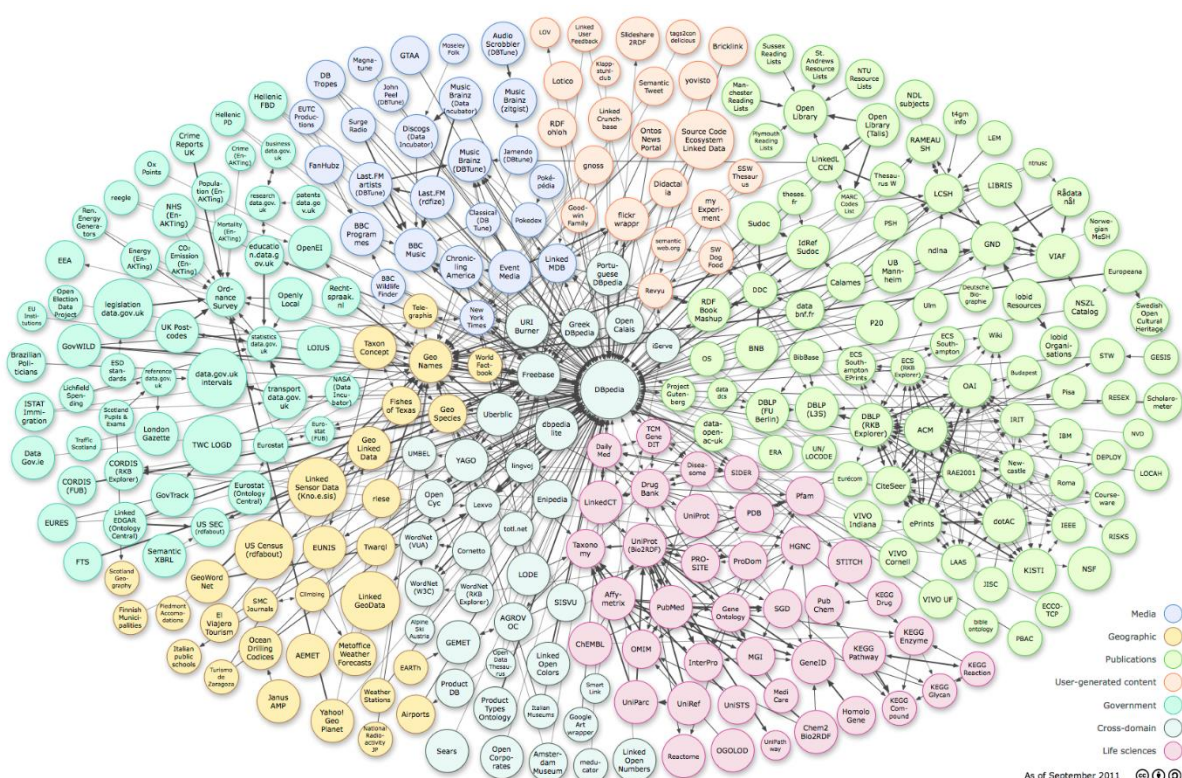


Figure 3.1.1 Image showing linked data connections across the World Wide Web

The Image above shows how different websites are using each other by utilising the semantic web. This shows the true power of the semantic web. To see a live and interactive version of the image above visit the following address <http://lod-cloud.net>.

3.2 Data Interchange - Resource Description Framework (RDF)

RDF is the backbone of linked data and it is what the World Wide Web Consortium (W3C) has recommended to create semantically linked data. RDF represents information about resources in a graph form, and since it is primarily intended for the www, it is built by representing resources with **Uniform Resource Identifier (URI)**.

URI's are used heavily in RDF. So, it important to understand the basics. The official definition from W3C is *"Uniform Resource Identifiers (URIs, aka URLs) are short strings that identify resources in the web. They make resources available under a variety of naming schemes and access methods"* (W3C, 2006).

So, entities (resources) are identified by URI's and use the *http://* scheme, so you can look up these entities simply by dereferencing the URI over the HTTP protocol.

RDF stores data and it is a file that should parse down to a list of triples. A triple consists of a *subject*, a *predicate* and an *object*. The *subject* identifies what object the triple is describing, the *predicate* defines the piece of data in the object you are giving the value to and the *object* is the actual value.

Tim Berners-Lee wrote an article called 'Linked Data – The Story So Far' (Berners-Lee, et al., n.d.) which describe the technologies used for the semantic web. In the article, he gave an example of RDF- *"an RDF triple can state that two people, A and B, each identified by a URI, are related by the fact that A knows B. Similarly, an RDF triple may relate a person C to a scientific article D in a bibliographic database by stating that C is the author of D. Two resources linked in this fashion can be drawn from different data sets on the Web, allowing data in one data source to be linked to that in another, thereby creating a Web of Data"*.

Here is what an RDF document may look like (see figure 3.2.1). The following code is taken from W3Schools (w3schools, n.d.).

```
<?xml version="1.0"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:si="https://www.w3schools.com/rdf/">

  <rdf:Description rdf:about="https://www.w3schools.com">
    <si:title>W3Schools</si:title>
    <si:author>Jan Egil Refsnes</si:author>
  </rdf:Description>

</rdf:RDF>
```

Figure 3.2.1 RDF file example

So, the above image shows a simple RDF triple. Now let's break it down to see the subject, predicate and object.

- Subject: 'https://www.w3schools.com'
- Predicate: 'title' and 'author'
- Object: 'W3Schools' and 'Jan Egil Refsnes'

This is a very simple triple because, in the true power of the semantic web, you could always use a resource URI as an object. So, in the example above, instead of writing the authors name as an object you could always have a URI pointing to the authors unique page, this would allow people to create more complex queries and gather more details about the author.

There is a great book written by Shelly Powers called 'Practical RDF'. In the book, she goes into incredible detail about RDF from what it is to how to create, use, and query it (Powers, 2003). I'd recommend reading it to get a deeper understanding of RDF.

3.3 Querying language - SPARQL

SPARQL (pronounced "sparkle") is an acronym for **S**imple **P**rotocol **A**nd **R**DF **Q**uery **L**anguage. It is the *"standard query language and data access protocol for use with the Resource Description Framework (RDF) data model"* (Microsoft, n.d.).

SPARQL consist of two parts. It is a query language and a protocol. SPARQL is used to query RDF data, in the same way, SQL is used to query relational data and XQuery is used to query XML data. But SPARQL is different because it is designed to operate over disconnected sources and the protocols allow the transmission of SPARQL queries and the results between a client and a SPARQL engine via HTTP.

The SPARQL language is simple to learn if you are already familiar with SQL because it shares several keywords such as 'SELECT', 'WHERE' etc. But it also has a few other new features that make the language powerful such as 'OPTIONAL' and 'FILTER'.

As I explained above (*section 3.2*) an RDF data file is created by one or more triple's and a triple consist of a subject, predict and object. So here is an example from Cambridge Semantics (Cambridge Semantics, n.d.). So, imagine we have the following RDF triples (*see figure 3.3.1*) –

```
ex:juan foaf:name "Juan Sequeda" .
ex:juan foaf:based_near ex:Austin .
```

Figure 3.3.1 RDF Triples

If we wanted to create a SPARQL query that returns every name from the dataset. Here is what it would look like (*see figure 3.3.2*) -

```

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM <http://example.com/dataset.rdf>
WHERE {
  ?x foaf:name ?name .
}
ORDER BY ?name

```

Figure 3.3.2 SPARQL query to get all names

When you first look at the above query you may notice it isn't that different to the RDF dataset and this is because SPARQL queries are based on the concept of *graph pattern matching* (W3C, 2005). So essentially a basic SPARQL query is a graph pattern with some variables, and the data returned will match the pattern in the query. In the example above (see figure 3.3.2) we have two variable (?x and ?name), and it has the foaf:name that matches the RDF dataset. After the query is sent the variable ?x would hold ex:juan and variable ?name would hold "Juan Sequeda".

In the above query (figure 3.3.2) there are a lot of different components that is used quite often such as PREFIX, SELECT, FROM, and WHERE.

PREFIX is used to simplify and minify the chances of errors in a query. It does this by specifying the namespace for the URIs. Here is an example of a query not using a PREFIX (see figure 3.3.3).

```

SELECT ?node ?name
WHERE{
  ?node <http://xmlns.com/foaf/0.1/givenname> ?name .
  ?node <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
}
LIMIT 10

```

Figure 3.3.3 SPARQL query without PREFIX

And here is the same query but using PREFIX's (see figure 3.3.4).

```

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?node ?name
WHERE{
  ?node foaf:givenname ?name ;
        rdf:type foaf:Person .
}
LIMIT 10

```

Figure 3.3.4 SPARQL query with PREFIX's

As you can see from the example above PREFIX's can make queries simpler to read and easier to understand. But it is important to remember that you don't need to use PREFIX's. It is just more convenient than rewriting the namespace each time.

Most SPARQL queries start with **SELECT** even though they can start with **ASK**, **DESCRIBE** or **CONSTRUCT**. The purpose of **SELECT** is very similar to **SQL** because it allows you to define which variables within the query you want to be returned, and just like **SQL** you can use the asterisk (*) symbol to specify that you want all the variable in the query returned.

The **FROM** keyword is used to specify what graphs you want to use. If you don't use the keyword **FROM** (or **FROM NAMED**) then when you send the SPARQL query to an endpoint, it will be executed against the default graph. *"The FROM and FROM NAMED keywords allow a query to specify an RDF dataset by reference"* (w3c, 2013).

The **WHERE** clause of the SPARQL query tells the SPARQL engine where to find the variables you defined in the **SELECT** clause. Following the **WHERE** clause is a pair of curly parenthesis ({}) where the main part of the query lies.

As discussed above a SPARQL query is very similar to the RDF dataset where it has a subject, predicate and an object. Within the SPARQL query the subject, predicate and object can be one of two things. It can either be a variable which is defined by putting a question mark (?) before the variable name or it can be a URI. When a subject, predicate and object is defined inside a SPARQL query it is then followed by a full stop (.), a semi-colon (;) or even a comma (,). If it ends with a full stop then another triple can be defined (*see figure 3.3.3*) but if you end with a semi-colon then the subject of the previous triple is used leaving only the predicate and object to be defined (*see figure 3.3.4*). And if you end with a comma (which is rarely used) then you don't have to define the subject or the predicate, only the object. These are useful because it reduces the chance of typos and makes the query simpler.

The following image (*figure 3.3.5*) shows how a SPARQL query is structured. As you can see it starts with the **PREFIX**'s which are optional, it is then followed by the query result clause (**SELECT**), then you define the dataset (**FROM** and **FROM NAMED**) which are also optional. Then you come to the main part of the query, the **WHERE** clause, where the main logic of the query is and where the variables are assigned values. At the very end of the query, you'll have the query modifiers which are optional. I won't discuss any of these as they are mostly self-explanatory.

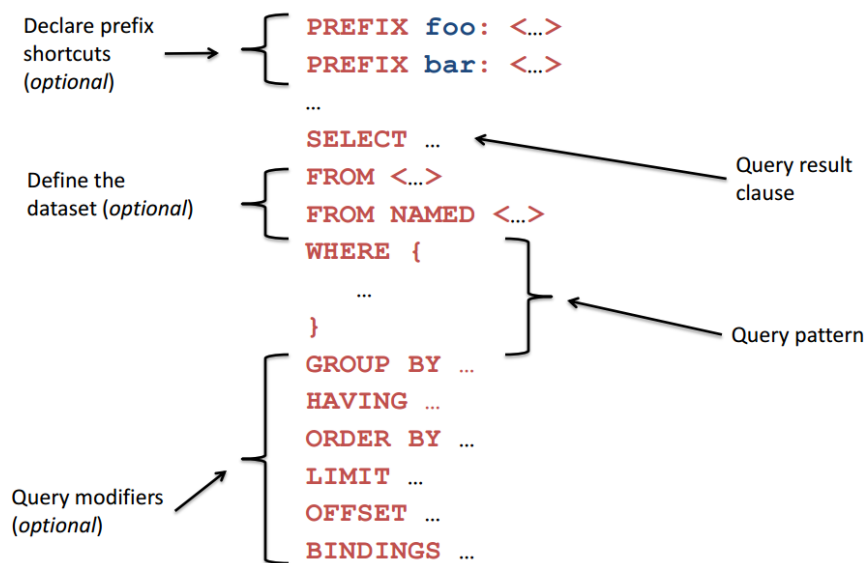


Figure 3.3.5 SPARQL query structure

Now there is only one big feature of SPARQL that I have not explained and this is **FILTER**. The FILTER clause is used within the WHERE curly parenthesis as a sub-clause. FILTER quite simply allows you to filter results on a certain condition. With the FILTER clause, you can filter by text, numbers and even dates. It is a very powerful feature that I will be utilising in my project.

SPARQL is a powerful query language that has a lot of useful features. SPARQL is specifically designed to query semantic data (RDF datasets) and is the most important query language for my project. To learn more about the individual components, I'd highly recommend the following website - <http://rdf.myexperiment.org/howtospargl>.

3.4 What is the difference between RDFS and OWL

When I first started to research RDFS (**R**esource **D**escription **F**ramework **S**chema) and OWL (**W**eb **O**ntology **L**anguage) it was quite confusing. As you can see from the figure 3.1 RDFS is the taxonomy and OWL is the ontology for the semantic web, and both are used together as part of the overall semantic web architecture.

But after some research, it became apparent that there is a big misconception of the two. For example, an article was written on 'Cambridge Semantics' titled 'RDFS vs. OWL' says *"We have introduced both RDFS and OWL as data modeling languages for describing RDF data. So which should you use?"* (Cambridge Semantics, n.d.). As you can see from the previous quote they are suggesting you can only pick one or the other.

But after quite a lot of research it has turned out that both are used to describe the data in the RDF data file, they just describe the data in different ways. Here is a great description of RDFS *"RDF itself simply creates a graph structure to represent data. RDFS provides some guidelines about how to use this graph structure in a disciplined way. It provides a way to talk about the vocabulary that will be used in an RDF graph. Which individuals are related to*

one another, and how? How are the properties we use to define our individuals related to other sets of individuals and, indeed, to one another? RDFS provides a way for an information modeler to express the answers to these sorts of questions as they pertain to particular data modeling and integration needs.” (Allenmang & Hendler, 2011)

And here is a description of OWL “The W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things.” (w3, 2009)

In a nutshell, “Each contains a set of defined terms, and each is critical to the ability to express the meaning of information. Their differences lie in their expressiveness, or how much meaning each attaches to the terms that it describes” (Hebeler, et al., 2009). See section 3.5 and 3.6 for more details.

3.5 Taxonomies – Resource Description Framework Schema (RDFS)

The Resource Description Framework Schema (RDFS), also known as RDF Vocabulary Description Language 1.0. RDFS is quite easy to write if you already understand RDF (see section 3.2) because it is directly written inside the RDF datasets and follows the same format.

What does RDFS do? Well, it is simply a set of classes and properties to provided more details about the data. It does this by defining **classes** and **properties**. Classes in RDFS are practically the same as in object orientated programming because you are able to define resources as an instance of a class or a subclass.

Dr Mustafa Jarrar has created a great slide show demonstrating RDFS (Jarrar, 2013). The following example of RDFS is from the slides of Dr Jarrar.

RDFS classes are described using RDFS resources, such as `rdfs:Class` and `rdfs:subClassOf`. These are used with the `rdf:type` property (see figure 3.5.1).

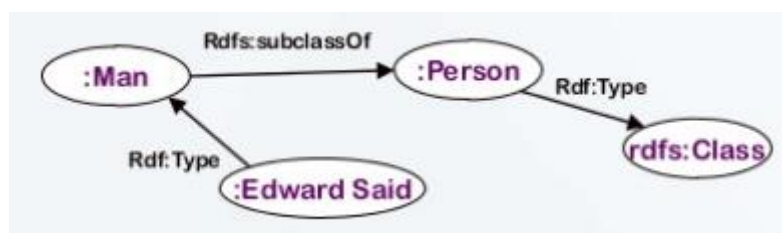
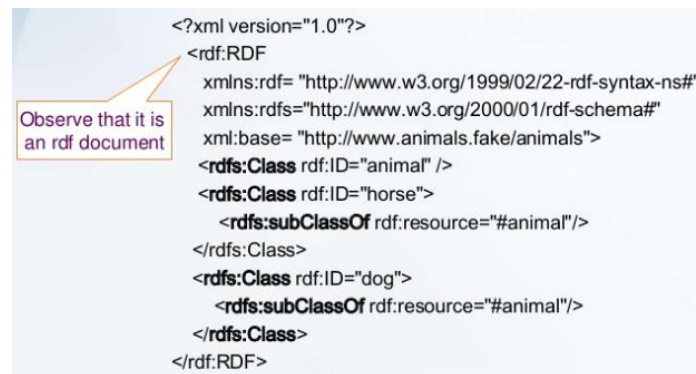


Figure 3.5.1 Example of RDFS structure

The above image (figure 3.5.1) is a diagrammatic example of how RDFS works. RDFS is written directly inside an RDF document (see figure 3.5.2).



```

<?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#"
  xml:base="http://www.animals.fake/animals">
  <rdfs:Class rdf:ID="animal" />
  <rdfs:Class rdf:ID="horse">
    <rdfs:subClassOf rdf:resource="#animal"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="dog">
    <rdfs:subClassOf rdf:resource="#animal"/>
  </rdfs:Class>
</rdf:RDF>

```

Figure 3.5.2 RDFS example in an RDF data file

Within the slides, he gives a brief description of a lot of the constructs within RDFS (see figure 3.5.3) but to learn more about RDFS I would suggest reading the official documentation from the ‘World Wide Web Consortium’ (W3C) <https://www.w3.org/TR/rdf-schema/>.

rdfs:Class allows to declare a resource as a class for other resources.
rdfs:subClassOf allows to declare hierarchies of classes.
rdfs:domain of an `rdf:property` declares the class of the *subject* in a triple whose second component is the *predicate*.
rdfs:range of an `rdf:property` declares the class or datatype of the *object* in a triple whose second component is the *predicate*.
rdfs:subPropertyOf is an instance of `rdf:Property` that is used to state that all resources related by one property are also related by another.
rdfs:seeAlso is an instance of `rdf:Property` that is used to indicate a resource that might provide additional information about the subject resource.
rdfs:label is an instance of `rdf:Property` that may be used to provide a human-readable version of a resource's name.
rdfs:comment is an instance of `rdf:Property` that may be used to provide a human-readable description of a resource.
rdfs:Literal is the class of literal values such as strings and integers. property values such as textual strings are examples of RDF literals. Literals may be plain or typed.
rdfs:Datatype is the class of datatypes...

Figure 3.5.3 RDFS constructs

3.6 Ontology – OWL

Owl operates on top of the RDF schema, by adding semantics to it. OWL is also expressed in triples like RDF and RDFS.

OWL allows you to specify a lot more details about the RDF data, such as, it can indicate that if ‘A isFriendTo B’ then ‘B isFriendTo A’. It can even take this logic even further, for example, if ‘A isAncestorOf B’, ‘B isAncestorOf C’ and ‘C isAncestorOf D’, then OWL can imply that ‘A isAncestorOf D’.

OWL also adds the ability to say that two things are the same, this is extremely useful if you are combining different RDF dataset, it does this with the predicate ‘owl:sameAs’. It is also useful if there are two datasets about the same thing, for example, ‘Liverpool FC’ ‘owl:sameAs’ ‘Liverpool Football Club’.

As you can see OWL is a powerful feature of the semantic web and is used in AI systems as reasoners.

3.7 Natural Language Processing (NLP)

I am not going to go into much detail about natural language processing as the bulk of my research is the semantic web. In recent years, Artificial Intelligence (AI) has become a big topic of discussion and research. AI is the simulation of human intelligence processed by machines. Some of these processes are learning, reasoning and self-correction. The term Artificial Intelligence was created by a computer scientist in 1956 called John McCarthy. AI encompasses a lot of different technologies such as pattern matching, automation, and robotics.

In this dissertation, the AI process I'm going to concentrate my project on is NLP, which stands for Natural Language Processing. *"NLP is a computational approach to text analysis"* (Jurafsky & Martin, 2009). As discussed in the introduction (see section 1), NLP *"is the automatic processing of text written in natural (human) languages (English, French, Chinese, etc.), as opposed to artificial languages such as programming languages, to try to "understand" it."* (Maynard, et al., 2016). NLP can also be known as Natural Language Engineering (NLE) or Computational Linguistics (CL).

In my project, I am going to implement an NLP to allow the users to search for items using natural language for example, 'List 5 films from 2015'. And the search engine will create a list of 5 films from 2015.

4. Methodology

This section discusses the research methodology for this research project. Several approaches are implemented in this research project to develop the software and to complete the objective of the research.

I will be using the research 'onion' developed by Saunders, et al., (2007) (see figure 4.1). I will be following Saunders research onion to clearly structure the methods that will be used in my research.

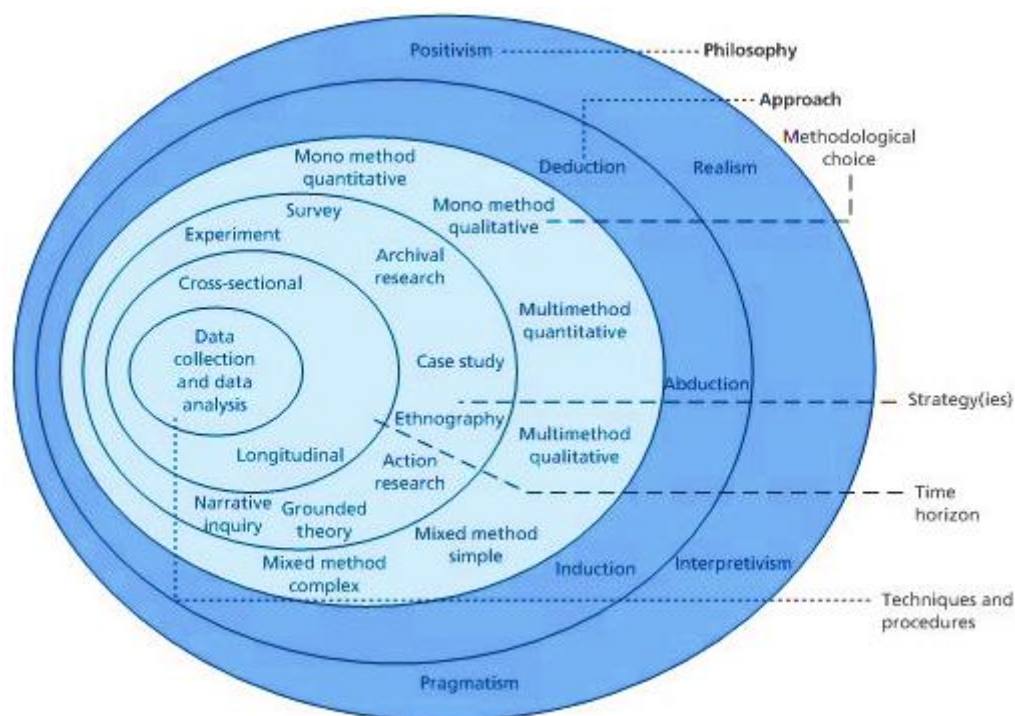


Figure 4.1 The research 'Onion'

4.1 Quantitative research

"Quantitative research gathers data in numerical form which can be put into categories, or in rank order, or measured in units of measurement. This type of data can be used to construct graphs and tables of raw data." (McLeod, 2088)

In my research, I will be collecting the time it took to process different searches to analyse the benefit of combining different technologies (traditional relational database and semantic technologies). By showing this I will be also partially answering one of my objective questions which is *"Investigate if semantic data can be combined with a traditional database"*.

This will be discussed in section 9 of the dissertation, where I discuss all my findings.

4.2 Design methodology

The design is obviously important since my dissertation contains a project. And to develop the application I had a lot of different decisions to make before the development of the project. Below is a table of all the choice's I made and I will discuss how I made the decisions below.

Choices	Programming Languages	Databases	Code Management	Natural Language Processor
1	C++	Oracle	The cloud	Wit.ai
2	JavaScript	SQL Server	Local Version Control Systems	Luis.ai
3	C#	MySQL	Centralized Version Control Systems	Api.ai
4	Java		Distributed Version Control Systems	
Selected option for this research:	C#	SQL Server	Distributed Version Control Systems	Luis.ai

4.2.1 Programming Language

Before starting the project, I had to decide what core programming language the application will be programmed in. So, the languages I considered were C++, JavaScript, C# and Java.

I chose not to use JavaScript almost immediately because it is very limiting for server side coding, except for the exception of Node.js, which left me to choose from C++, C# and Java.

I decided against C++ for many reasons such as it has a limited library compared to C#, and it doesn't have a native garbage collection compared to Java and C#. A *"garbage collector manages the allocation and release of memory for your application"* (Microsoft, n.d.).

So, my final choice was between Java and C#. At first, I was leaning more towards Java because throughout my years of programming I have dominantly used Java. Also originally the project was going to be completely console based which means it *"is an application that takes input and displays output at a command line console"* (techopedia, n.d.). In other words, it would have no user interface. But then I decided to create a web application that would be user-friendly to show the full power of the semantic web.

I ended up choosing C#. And this was a tough choice because *"both have extensive libraries that can be used to build applications for the desktop, web, mobile devices, and other platforms. Both have large communities of enthusiastic fans, and plenty of online support."* (Shiotsu, n.d.)

But I ended up picking C# because it has a few features that would make the development of the project simple such as LINQ, LINQ is a query language that makes it simple to query a data source, and will also a powerful web framework called ASP.NET.

4.2.2 Databases

The application requires a database to store the returned results from an SPARQL query. *“Computer database in which all data is stored in Relations which (to the user) are tables with rows and columns..... Every table shares at least one field with another table in 'one to one,' 'one to many,' or 'many to many' relationships. These relationships allow the database user to access the data in almost an unlimited number of ways, and to combine the tables as building blocks to create complex and very large databases. See also flat database.”* (Business Dictionary, n.d.)

So, I wanted a relation database system that is widely used. Therefore, I had to decide between Oracle database, SQL server and MySQL.

I decided not to use Oracle database because as I stated up I will be using C# and ASP.NET and Oracle database works best with Java and is quite difficult to incorporate into an ASP.NET web application.

So, I was left with SQL Server and MySQL. This was a tough choice as they are two of the most popular relational database management systems on the market and it isn't clear which one is the best.

I have decided to use SQL Server. Both are very similar and even though MySQL is open source and free to use, SQL Server is integrated into ASP.NET and it offers higher security than MySQL because as it states in an article from 'the windows club' SQL server is easier to keep up to date (The Windows Club, n.d.).

4.2.3 Code Management

Before I start the project, I had to decide how I'd manage (store) the code. I obviously had several options such as the Cloud, Local Version Control System, Centralized Version Control System, and Distributed Version Control System.

I immediately chose not to use the Cloud because it offers very little benefits to the coding experience and is has very limited functionality.

So now I was left with the 3 different Version Control Systems. I am going to explain what a version control system is and then what are the differences between the three of them and discuss my final decision.

So, what is version control? *“Version control systems are software that helps you track changes you make to your code over time. As you edit to your code, you tell the version control system to take a snapshot of your files. The version control system*

saves that snapshot permanently so you can recall it later if you need it.” (Outlaw, n.d.).

There are many benefits to version control such as it keeps a history of changes, and from the history, you can find out who, why and when the changes were made. By keeping a history, it gives developers the confidence to experiment because they can always revert back to a previous version. By using a version control system, it means every version has a description of what the changes were, such as, fixed a bug or added a new feature. This makes it very easy to follow the changes made and to pick a version to go back too.

4.2.3.1 Local Version Control System

Local Version Control Systems is a simple database that keeps all the changes to files under version control. As the term ‘local’ suggests this will only be on your system. The image below demonstrates the simple flow of local version control systems.

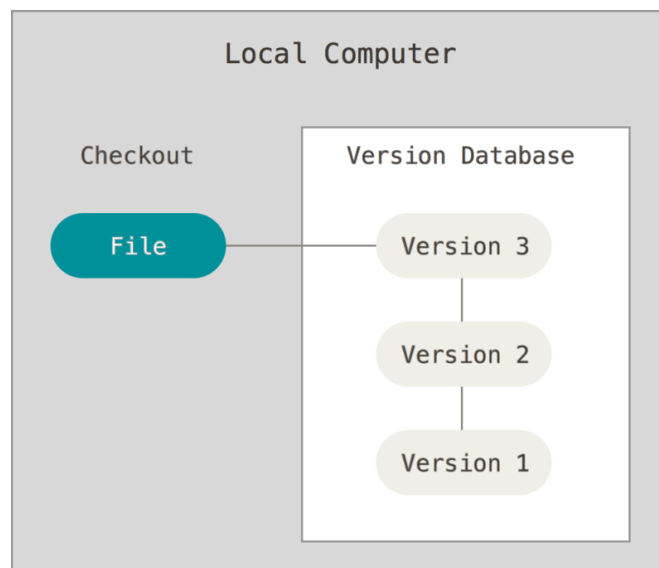


Figure 4.2.3.1.1 How Local Version Control Systems work

But the major issue people encountered was that they needed a way to collaborate with other developers, which is why the centralised version control system was developed.

4.2.3.2 Centralised Version Control System

Centralised version control system features a main server that contains the versioning files and then a client can check out files from a central place.

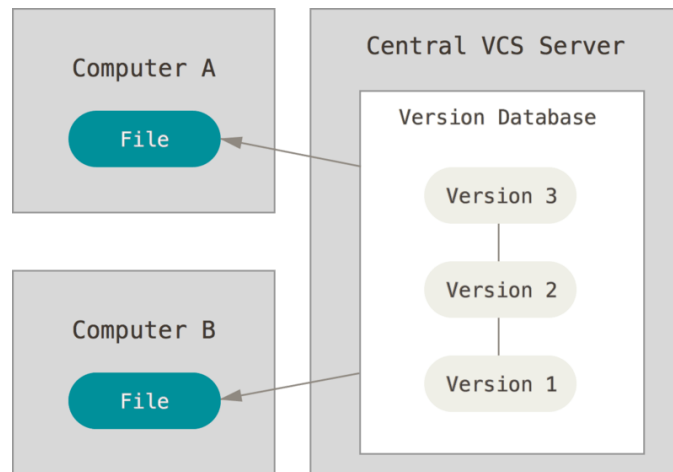


Figure 4.2.3.2.1 How Centralised Version Control Systems work

There are a lot of benefits to this system over the local one. It allows easier collaboration because any number of developers can retrieve a file and commit the changes and it is easier to administer because you have more control of who can do what.

But there is still a major downside. If the *“server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they’re working on. If the hard disk the central database is on becomes corrupted, and proper backups haven’t been kept, you lose absolutely everything – the entire history of the project except whatever single snapshots people happen to have on their local machines. Local VCS systems suffer from this same problem – whenever you have the entire history of the project in a single place, you risk losing everything.”* (git-scm, n.d.)

4.2.3.3 Distributed Version Control System

Now we are at the latest stage called Distributed Version Control Systems. As you can see from the image below (see figure 4.2.3.3.1) unlike a centralised system where a user just downloads the latest snapshot of the files, on a distributed version control system a user clones/ mirrors the entire repository, meaning if anything happened to the central server each client who downloaded/ cloned it has a full backup that can be copied back to the server if there was a failure.

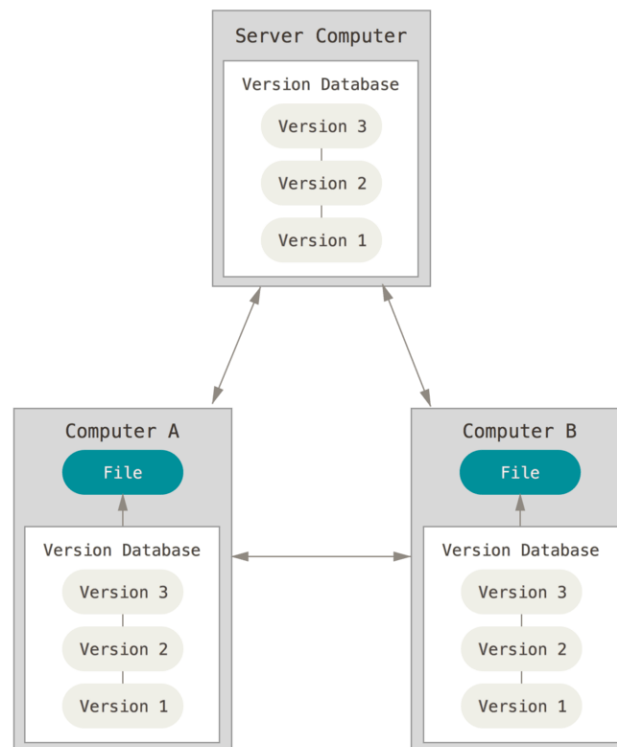


Figure 4.2.3.3.1 How Distributed Version Control Systems work

As you can see above I have done a lot of research into Version Control Systems, and for my project, I will be utilising the latest version which is a Distributed Version Control System. The reason I have chosen this approach is because there is a lot more security because there will be multiple copies of my project meaning the risk of losing the project is very low.

I will be using a website/ application called GitHub, which is a well know distributed version control systems.

4.2.4 Natural Language Processor

A big part of the application is the Natural Language Processor. So, I had to find an API that was easy to use and easy to implement. When I first started research different API's I first came across Luis.ai which is developed by Microsoft so it will be compatible with C#, SQL server and ASP.NET. But to ensure I pick the right one for my project, I found two other API's. The choices are Wit.ai, Luis.ai, and Api.ai.

To help make the choice I created a table with all the relevant information.

Wit.ai	Luis.ai	Api.ai
Completely free to use	Free to use up to 10k transaction month	Free to use with optional enterprise option
Combination of voice recognition and machine learning	Uses machine learning to analyse sentences	Speech to text and Text to speech capabilities along with machine learning
Owned by Facebook	Owned by Microsoft	Owned by google

As you can see the difference are not great, I stuck with my 'gut' instinct and went with Luis.ai because it lowers the chance of compatibility issues and they provide guides of integrating it with C#.

4.2.5 Conclusion

So, in this section I have discussed the core design choice and why I made them. As a quick recap, I will be using C# as my main logic language along with ASP.NET MVC as a framework to build the web application with. The database will be built using SQL Server and the natural language processing will be handled by Luis.ai. The entire project will be managed and backed up using Distributed Version Control System more specifically GitHub.

5. Project Planning

I have from early September 2016 to the 24th of April 2017 to complete the entire research. I plan to have 3 separate stages. In the first part, I plan to do some research (*see section 5.1*) to get a clear idea of my topic. The second stage will be the development of my project (*see section 5.2*) and the final stage will be a discussion (*see section 5.3*) where I will outline all my findings and conclusions.

To make sure that I am on track for completing the project/ research, I will use several techniques to monitor my progress. For example, I will schedule regular meetings with my supervisor to keep him updated on my progress. I have also created a detailed Gantt Chart using Microsoft Project to follow during the project. Using this Gantt chart will help me immensely to monitor the progression of my dissertation. Here is an image of the Gantt chart (*see figure 5.1*).

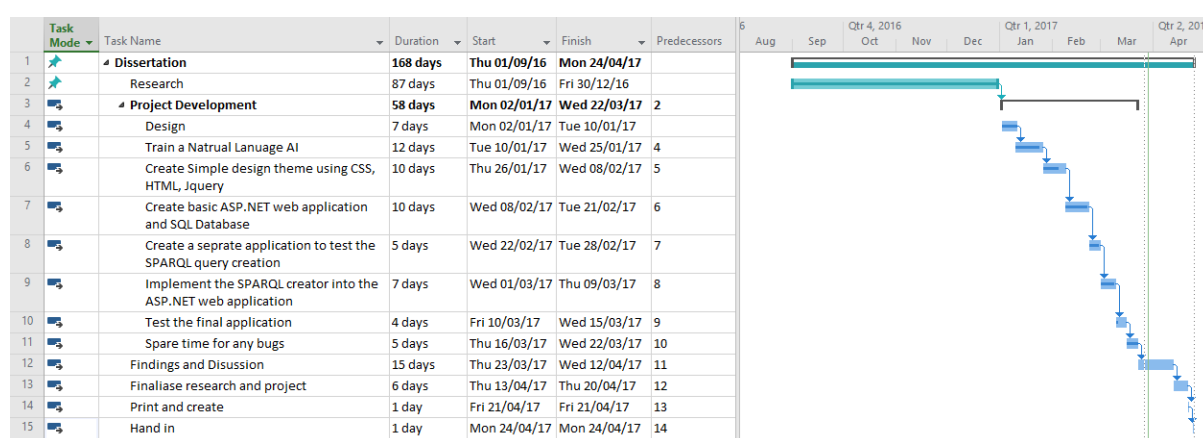


Figure 5.1 Gantt Chart Created in Microsoft project 2016

5.1 Research Period

I will start the research period from September 2016 until the end of December 2016. During this time, I plan to find out more about semantic web technologies, techniques, and languages. Doing the research before the development will hopefully answer a lot of the questions I mentioned earlier and give me further insights and ideas into the application I will be developing.

The main part of this research period will be writing a literature review where I will discuss other research related to my topic area.

Once I am confident that I have conducted enough research into the topic area I will then start to develop the application.

5.2 Development

As you can see from the Gantt chart, I plan to start the development of the project around half way through the time set for the dissertation. The reason I will start the project half way through the allocated time frame is so that I have a good understanding of the technologies involved to design a high standard web application.

When developing the web application, I plan to follow an evolutionary approach more specifically 'Scrum'. 'Scrum' is a system development life cycle based on the agile manifesto. *"A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value."* (Scrum Guides, n.d.). The scrum involves creating a backlog of user stories (tasks) and these user stories are separated into sprints, which usually lasts 2-3 weeks. After each sprint, there will be a working piece of software. Before the next sprint starts the prototype from the earlier sprint is evaluated. At the end of all the sprints (iterations), there will be a complete program.

This means throughout the development I will create several prototypes before the final application is created. The main reason I will follow this approach is because it will allow me to make changes to the software as I develop it and if the requirements change. The first prototype will have very little functionality. I will then test the prototype to see what can be improved and if the requirements need to be changed. I will then repeat this stage. Building on top of the previous prototype. I will conduct as many iterations that are needed until all the requirement are met.

There will then be the final iteration where my main priority will be finding any functionality problems and bugs. Utilising this process will allow me to have a working program as early as possible, and through each iteration, the program will improve. I will perform as many iterations that are needed to complete the project at a high standard.

I plan to complete the development by the end of March. This should give me plenty of time to complete the development and I should hopefully have enough time after development to write about the project. Also by giving myself this extra time at the end of the project will hopefully mitigate any risk if the development estimation needed to be increased.

5.3 Discussion and findings

This is the final part of my dissertation, and it is where I will discuss all my findings. As you can see from the Gantt chart I will have roughly two weeks to write this section.

This section is important because it is where I will answer all my questions and address the purpose of this dissertation.

6. Requirements and analysis

The software I am creating to support my research is a simple web application that uses the Semantic web and Natural Language Processing.

The application will accept a string and use Natural Language Processing (NLP) to determine what the user is asking for and then use semantic web technologies to gather the relevant information and display it. In the next subsection, I will be discussing the technology I will be using.

6.1 Languages and technologies

The application is going to be developed in ASP.NET using C#.

“ASP.NET is an open source web framework for building modern web apps and services with .NET. ASP.NET creates websites based on HTML5, CSS, and JavaScript that are simple, fast, and can scale to millions of users.” (asp.net, n.d.).

The web application will utilise all the modern and current technology/ languages and techniques such as MVC, C#, SQL, SPARQL, JQuery, HTML5 and SASS.

I will be using C# and ASP.NET framework for the logic of the application, for example, it controls the process of a user's request. I have explained my decision above (*section 4.2.1 Programming Language*) for why I am using these two technologies.

SPARQL is the query language I will be using to retrieve the data from [DBpedia](#). SPARQL is a semantic RDF query language able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. So, in my application, I use a SPARQL query to retrieve data from [DBpedia](#) and store a local record of the results in a relational database.

And finally, I will use multiple languages for the look and feel of the website. For example, I use HTML5 for the structure of the website, [JQuery](#) which is a JavaScript library to add interactivity to the website and [compass](#) which is a framework for [SASS](#) which is an extension to CSS3 to style the web application and to make it responsive.

6.2 Natural Language Processing

A big part of my dissertation is NLP (Natural language Processing). To incorporate this functionality into the web application, I will be using an API called LUIS.

LUIS stands for **L**anguage **U**nderstanding **I**ntelligent **S**ervice, and according to the website ([LUIS.ai](#)) LUIS offers a fast and effective way of adding language understanding to applications. It was introduced during Microsoft Build 2016 in San Francisco.

Luis is very easy to understand, *“it's a Machine Learning API that allows you to create an app with some intents and entities and then train it on some utterances”* (Alghazawi, 2016).

So, Luis works by calling an HTTP endpoint with a query (text input) and it returns a result (matching 'intent' with a 'confidence' score, as well as any extracted 'entities') in the form

of a JSON file. So, what are intents and entities? Well it's simple intents is an action and the entities are the parameters. See the image below (*figure 6.2.1*) for better understanding.

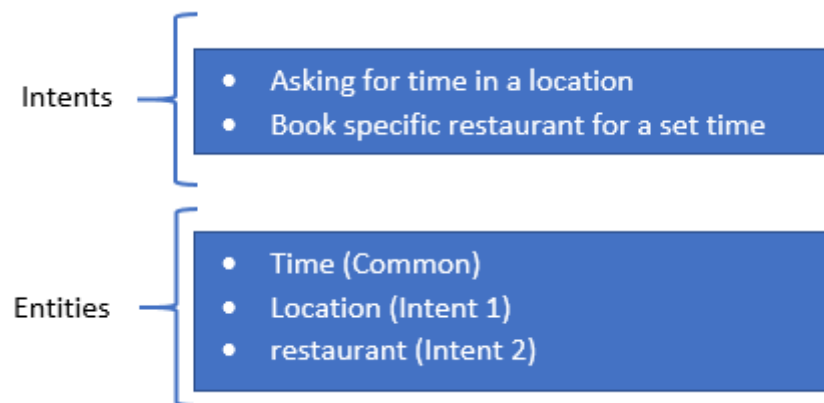


Figure 6.2.1 Image of intents and entities

In the implementation section of this dissertation, I will show how I used the Luis.ai and how I implemented it.

6.3 DBpedia

DBpedia was officially launched in 2007 at the World Wide Web conference in Banff. It is the web accessible RDF model database derived from Wikipedia. The data on DBpedia gets updated from Wikipedia periodically. DBpedia is an RDF database which is a linked data space comprised of HTML+RDFA based data and a SPARQL endpoint.

DBpedia is very important because it is a complementary service to Wikipedia by exposing Wikipedia-knowledge in a computer readable format. And an article on 'Medium' (Idehen, 2016) explains it has helped companies such as Apple, Google, and IBM to enhance their projects that involve AI.

So, in my application, I will be using DBpedia as my data source. I will utilise the SPARQL endpoint that they offer and process the returned data.

7. Design

In this section, I will design the application and explain the design choices I have made.

7.1 Design Patterns

The design of my application is going to be very simple and follow a traditional MVC design. MVC is a design pattern. What are design patterns? Design patterns are solutions to general problems that software developers have faced. Thinking about design patterns before you start your project is important because these patterns will solve many problems you could come across. These patterns have been developed over many years of trial and error.

*“The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the **model**, the **view**, and the **controller**. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.”* (Tutorials Point, n.d.)

7.2 Database

To answer one of my objectives (“Investigate if semantic data can be combined with a traditional database”), I am going to have to create a SQL database and store the returned data from the semantic web. Here is the database design (see figure 7.2.1).

MovieUserSearch		
PK	ID	int
	SearchedFor	string
	MovieLink	string
	Title	string
	GenreLink	string
	Genre	string
	ReleaseDate	string
	LastUpdated	Date Time

Figure 7.2.1 Database design

The design above will be implanted into the ASP.NET web application using entity framework code first approach. I explain this in section 8.6 where I show I created and developed the database.

7.3 Search process

So, the whole application is based on how the search is processed so it's important to design and understand the search process before implanting it. Here is a diagram I create to model the search process (*figure 7.3.1*).

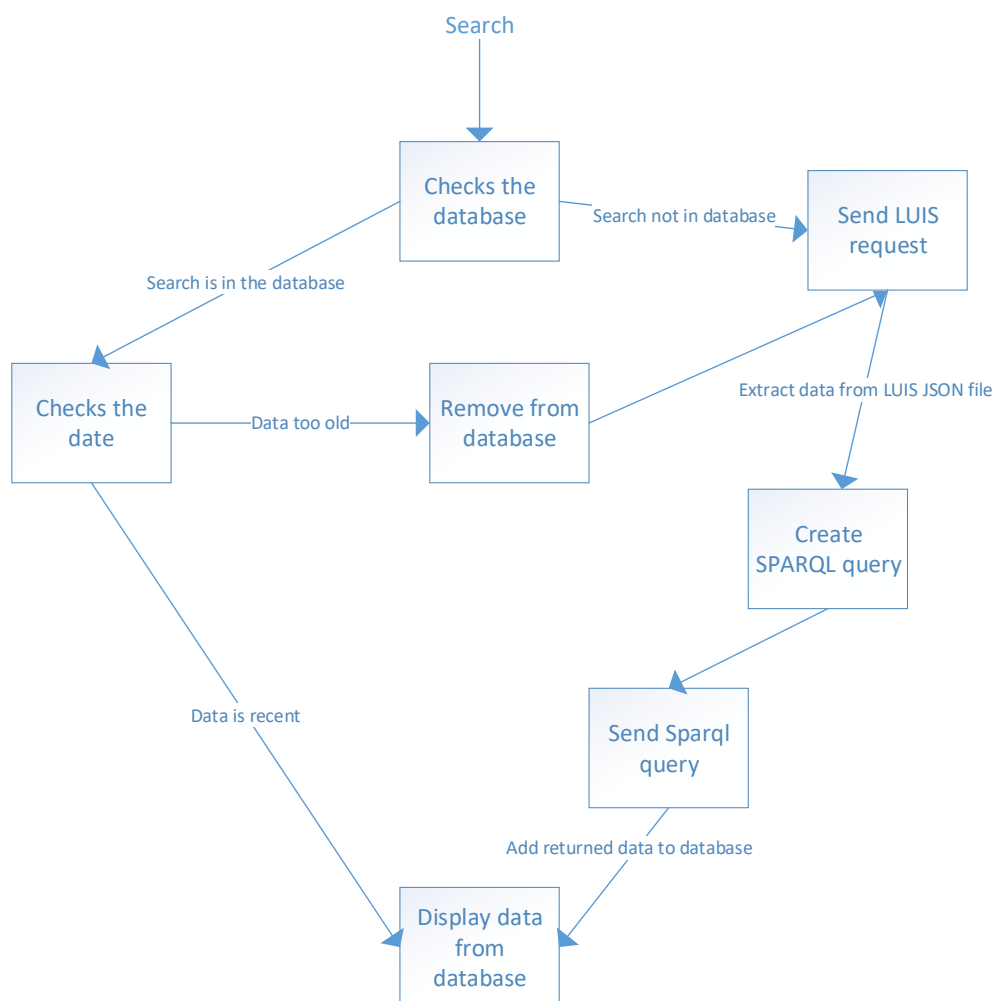


Figure 7.3.1 Image showing the search process of the application

The diagram above is a simple representation of how the search will be processed. Once the search is initiated it checks if it has been searched for before. If it has been it will then check when the data was last updated. It then has two options –

1. If the information doesn't need updating, then the data from the database will be displayed for the user.
2. On the other hand, if the data needs to be updated there are a few more steps to take. It will first remove all the old data from the database, then it will send the

search query to LUIS, the data from LUIS gets extracted to create a SPARQL query which is then used to request the data from DBpedia. Once the results from DBpedia is processed and added to the database, it will then be displayed for the user.

This search process is relatively simple and it is designed to be quick and efficient. I explain why in more detail below (*see section 7.3.1*).

7.3.1 Optimisation based on time of search to reduce calls across the network

I have optimised the web application to reduce the number of calls across a network. This will be done by utilising a SQL server database to store results from search queries. Designing the application this way has two benefits

The first benefit is that users should notice an increase in speed if the results are already in the database and don't need updating because it will simply display the results from the database.

And the second benefit is that it will reduce the number of calls across the network. What this means is that it will only query LUIS and DBpedia if it needs new data. This is because all the data needed would be stored locally.

8. Implementation and testing

This section I will show the process of integrating all these technologies and how I developed the application.

8.1 Designing the user interface

To design and create the user interface I used several different languages. The languages I used were HTML5, JQuery, Compass, and Bootstrap.

Before I started the design, I had to think of what was needed. And to be honest it was a simple application to design because all it requires is a home page with a search bar and a page to display the search results. The whole design will be responsive and modern.

Here is the final design of the application.



Figure 8.1.1 Home page design

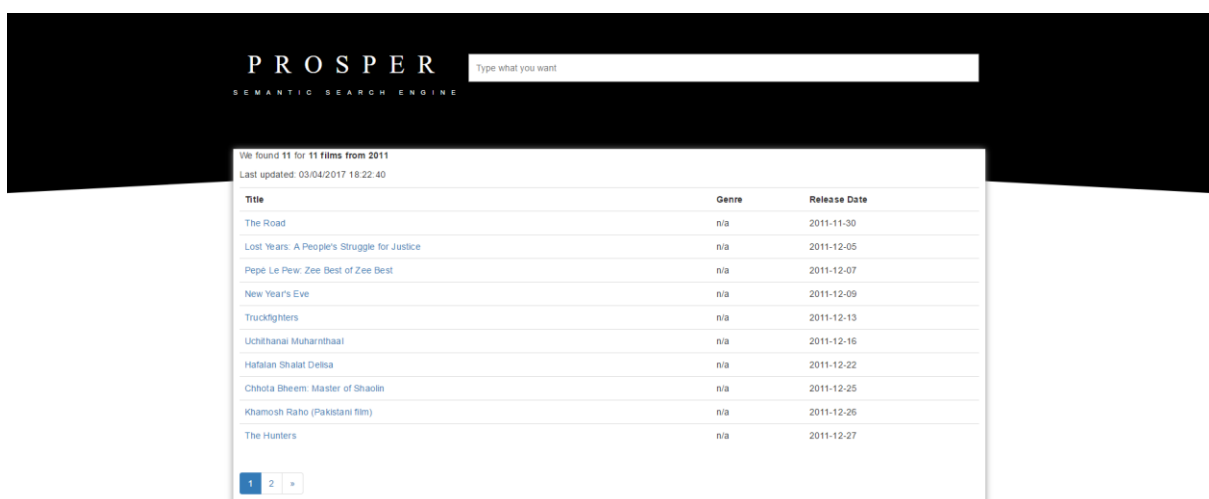


Figure 8.1.2 Search result page design

As you can see from the two image above (*figure 8.1.1 and figure 8.1.2*) I have kept the design minimal and simple.

8.2 LUIS

Creating a LUIS app is quite simple and on the LUIS website they have divided it into 4 stages, these are - Build your app, improve and tweak, test your app, and publish your app.

8.2.1 Build you app

So obviously the first step is to create a Luis App.

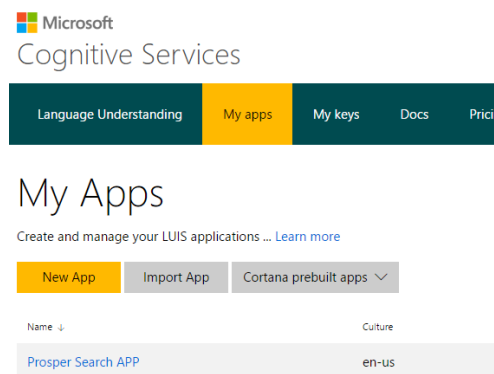


Figure 8.2.1.1 Image of app creation

Once the app is created, I had to create the app intents. As discussed above (*section 6.2*) an intent is an action such as, 'get weather' or 'remind me when ...'. In my application, I will have one intent which is 'movie search'. The image below (*figure 8.2.1.2*) shows the intent created.

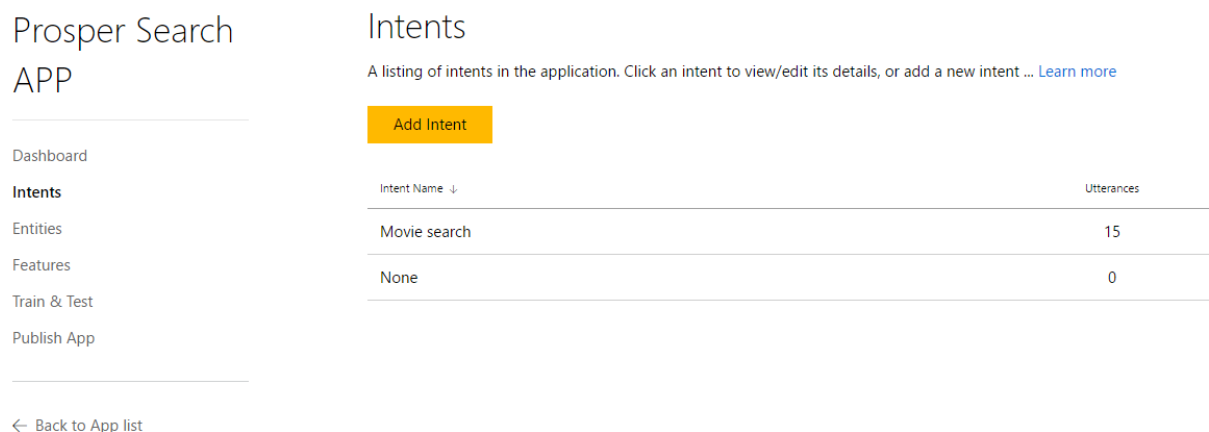


Figure 8.2.1.2 Image of intents

After adding an intent, I needed to decide what entities I will need. So, I am only going to create a simple movie search query, where I need to collect 3 different pieces of information such as 'number of returned results', 'genre' and the 'date'.

Fortunately, Luis provides prebuilt entities that Luis has already been trained to recognise. The prebuilt entities I used were 'number' and 'DateTime'. So, I only had to create one more for the genre. You can see the entities in the image below (*figure 8.2.1.3*).

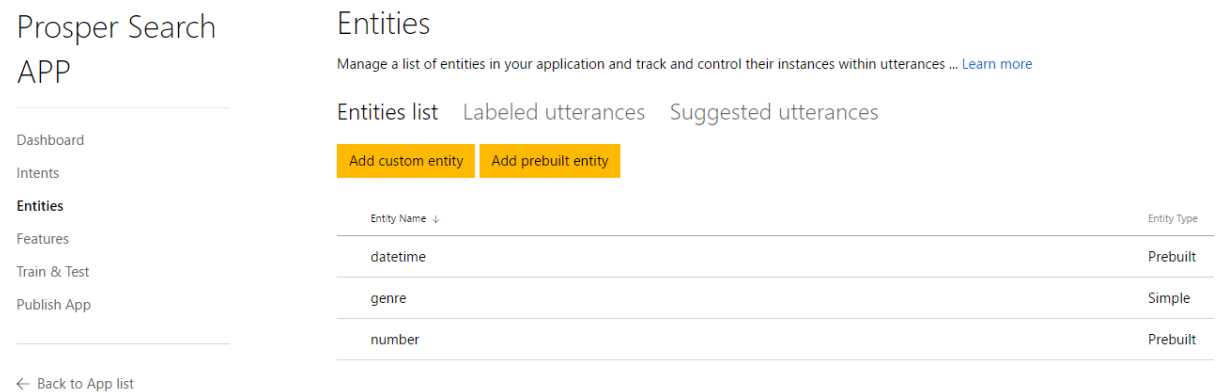


Figure 8.2.1.3 Image of entities

Once I created the entities it was time to train Luis to spot them by labelling possible utterance (sentences). By creating utterances and labelling the entities within, Luis will be able to learn from them. As you can see from the image below (*figure 8.2.1.4*) you should feed Luis as many possible utterances as possible to ensure an accurate result.

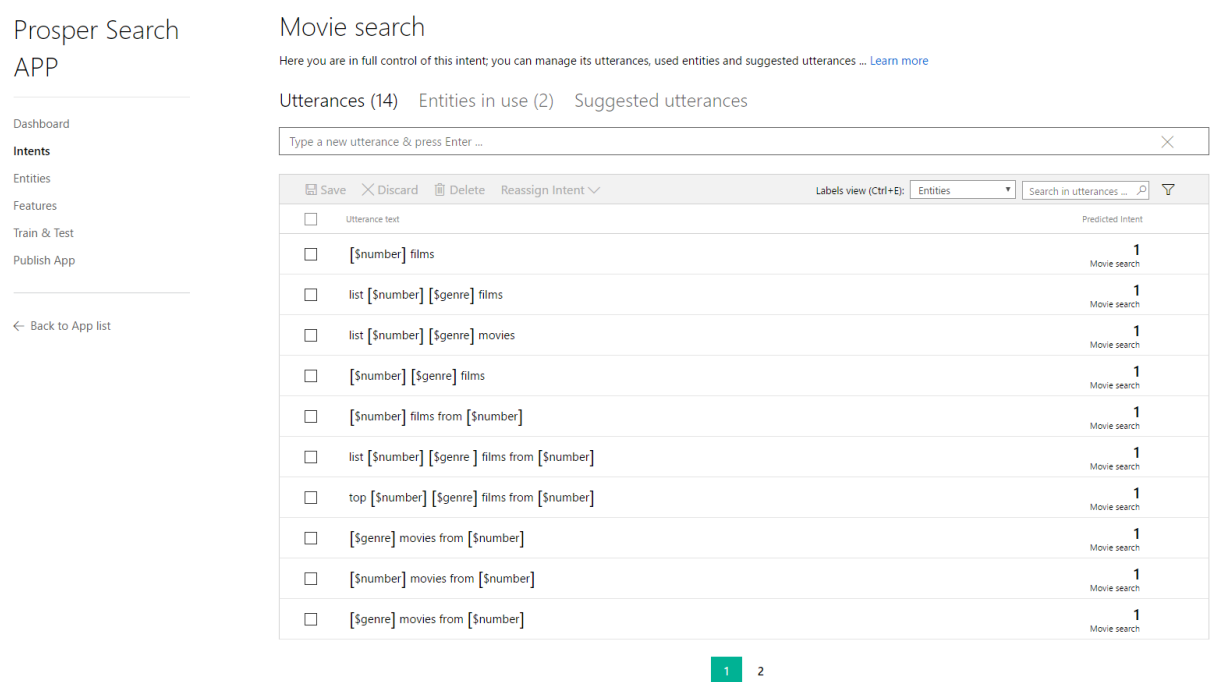


Figure 8.2.1.4 Image showing utterances for an intent

8.2.2 Improve and tweak

At this point, I have created a simple Luis application, but Microsoft has also provided a way to improve it by using powerful features such as phrase lists and a regex feature. For my application, I only needed to use a phrase list. The phrase list will contain a list of possible genres. The possible uses for a regex would be identifying product number or banking details.

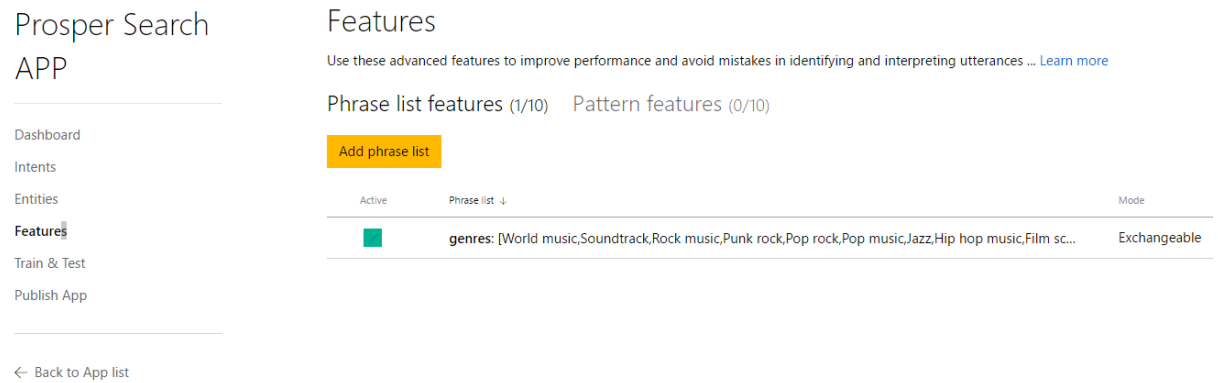


Figure 8.2.2.1 Image of a phase list

8.2.3 Test your app

Once I completed the above steps it was time to test the Luis application. And Microsoft has provided a way to 'Train & Test' your Luis application. Below is a screenshot of the testing I conducted.

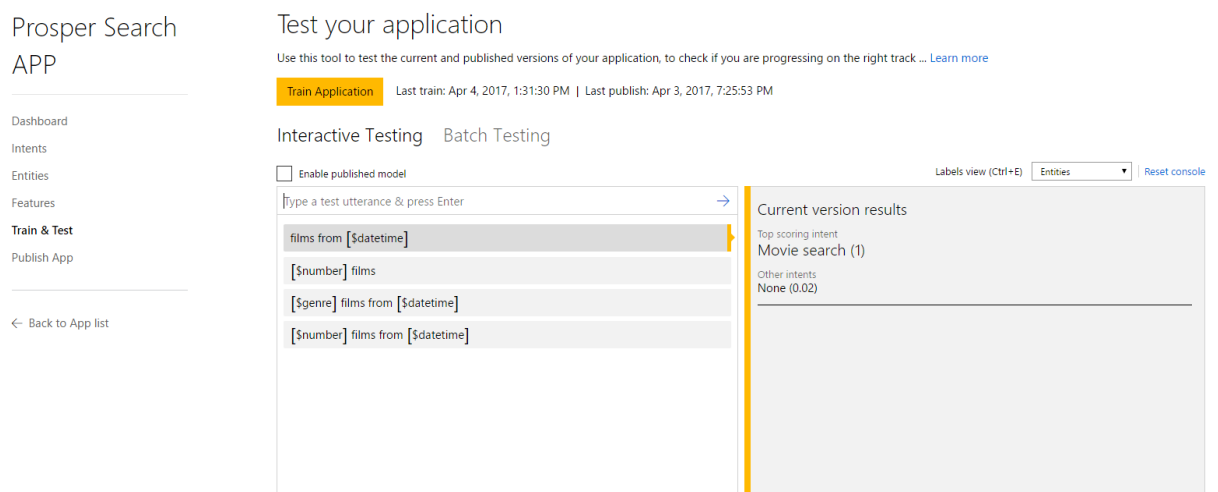


Figure 8.2.3.1 Image of the tests conducted against the Luis app

8.2.4 Publish your app

After I conducted the testing and there were no errors, it was time to publish the Luis application. As you can see from figure 8.2.4.1 once the app is published you are provided with a HTTP endpoint with the Luis application id and your subscription key.

Publish App

← Back to App list

Publish settings

Endpoint slot
Production

Slot info
Published version Id: 0.1
Published date: Apr 3, 2017, 7:25:53 PM

Endpoint url
https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/...?subscription-key=...&verbose=true&q=

☒ Add verbose flag ☐ Enable Bing spell checker

Timezone
(GMT) Western Europe Time, London, Lisbon, Casal

Train Publish

Figure 8.2.4.1 Luis application begin published

8.3 Creating SPARQL query

Creating the SPARQL query is one of the main parts of this project because without the correct SPARQL query the project won't work. Here is the SPARQL query I created (see *figure 8.3.1*).

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX db: <http://dbpedia.org/ontology/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT ?movieLink ?title ?genreLink ?genre ?releaseDate
WHERE {
  ?movieLink rdf:type db:Film;
    foaf:name ?title.
  OPTIONAL { ?movieLink prop:genre ?genreLink.
    ?genreLink rdfs:label ?genre.
    FILTER(lang(?genre) = 'en') }.
  OPTIONAL { ?movieLink <http://dbpedia.org/ontology/releaseDate> ?releaseDate }.

  FILTER ( regex (str(?genre), 'soundtrack', 'i'))
  FILTER ((?releaseDate >= '2010-01-01'^^xsd:date) && (?releaseDate < '2010-12-31'^^xsd:date))
  FILTER(lang(?title) = 'en')
}
ORDER BY DESC(?releaseDate)
LIMIT 100
```

Figure 8.3.1 SPARQL query created for this project

As you can see it uses a lot of the features discussed in section 3.3, such as PREFIX, FILTER, ORDER BY, and LIMIT. I needed to create a SPARQL query that could query for a specific genre, a year and return a certain number of results. I achieved this by using a combination of FILTERS and using the LIMIT as a query modifier at the end.

I explain in section 8.4 how the query gets modified for the user's input.

8.4 Generating a SPARQL Query

To generate a custom SPARQL query I needed to use the returned data from Luis. So, I created a simple console version before implementing it into the ASP.NET web application.

So, the simple application consisted of 3 classes, u can see this in *figure 8.4.1* –

1. Utilities – which includes several methods
2. LuisJSONModel – This class is the model of the JSON file returned from LUIS.
3. Program – This is the main class of the program.

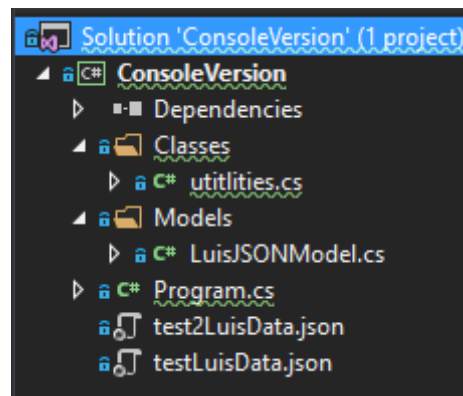


Figure 8.4.1 Program structure

The first step was to create a class that would model the returned JSON file from Luis. To do this, I needed to understand the structure of the JSON file. After analysing the file, I create a class called LuisJSONModel.

```

1 reference | Ieuan Walker, 12 days ago | 1 author, 1 change
public class TopScoringIntent
{
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public string intent { get; set; }
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public double score { get; set; }
}

1 reference | Ieuan Walker, 12 days ago | 1 author, 1 change
public class Intent
{
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public string intent { get; set; }
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public double score { get; set; }
}

1 reference | Ieuan Walker, 12 days ago | 1 author, 1 change
public class Resolution
{
    1 reference | Ieuan Walker, 12 days ago | 1 author, 1 change
    public string value { get; set; }
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public string comment { get; set; }
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public string time { get; set; }
}

1 reference | Ieuan Walker, 12 days ago | 1 author, 1 change
public class Entity
{
    4 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public string entity { get; set; }
    1 reference | Ieuan Walker, 12 days ago | 1 author, 1 change
    public string type { get; set; }
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public int startIndex { get; set; }
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public int endIndex { get; set; }
    1 reference | Ieuan Walker, 12 days ago | 1 author, 1 change
    public Resolution resolution { get; set; }
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public double? score { get; set; }
}

6 references | Ieuan Walker, 12 days ago | 1 author, 1 change
public class LuisJSONModel
{
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public string query { get; set; }
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public TopScoringIntent topScoringIntent { get; set; }
    0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
    public IList<Intent> intents { get; set; }
    1 reference | Ieuan Walker, 12 days ago | 1 author, 1 change
    public IList<Entity> entities { get; set; }
}

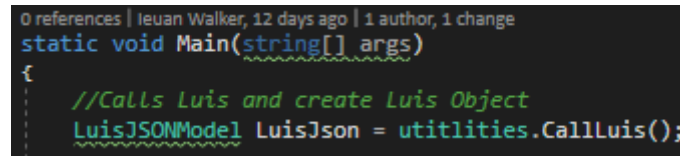
```

Figure 8.4.2 Luis JSON model class

Since this is a simple test application, I decided to download two test JSON files that are similar to what could be returned from a request to Luis. You can see this in the image

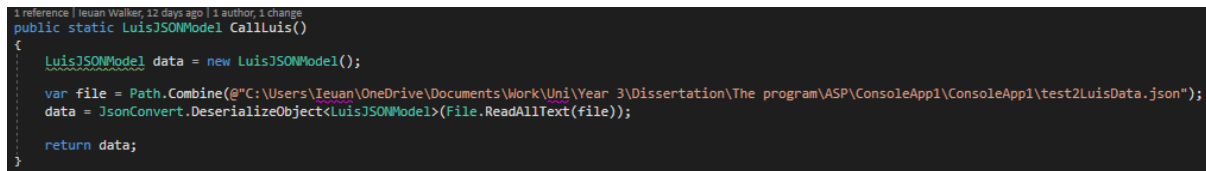
above (see figure 8.4.1). There are two JSON files in the root of the application ('testLuisData.json' & 'test2LuisData.json').

So when the application first starts it calls a method called CallLuis(). This method simply reads one of the JSON files, converts it to an object using LuisJSONModel class and returns the object. You can see this in the images below (figure 8.4.3 and figure 8.4.4).



```
0 references | Ieuan Walker, 12 days ago | 1 author, 1 change
static void Main(string[] args)
{
    //Calls Luis and create Luis Object
    LuisJSONModel LuisJson = utilities.CallLuis();
}
```

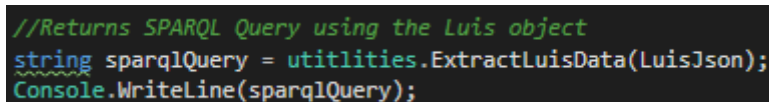
Figure 8.4.3 Calling the method CallLuis() from the main program (program.cs)



```
1 reference | Ieuan Walker, 12 days ago | 1 author, 1 change
public static LuisJSONModel CallLuis()
{
    LuisJSONModel data = new LuisJSONModel();
    var file = Path.Combine(@"C:\Users\Ieuan\OneDrive\Documents\Work\Uni\Year 3\Dissertation\The program\ASP\ConsoleApp1\ConsoleApp1\test2LuisData.json");
    data = JsonConvert.DeserializeObject<LuisJSONModel>(File.ReadAllText(file));
    return data;
}
```

Figure 8.4.4 The CallLuis() method

Once the LuisJSONModel object is returned it is then passed to the method ExtractLuisData(), where it returns the SPARQL query as a string (see figure 8.4.5).



```
//Returns SPARQL Query using the Luis object
string sparqlQuery = utilities.ExtractLuisData(LuisJson);
Console.WriteLine(sparqlQuery);
```

Figure 8.4.5 Calling ExtractLuisData() method

When ExtractLuisData() is called there are two methods that must be used before it returns the SPARQL (ExtractLuisData() and CreateSparqlQuery()). The first method (ExtractLuisData()) simply iterates through all the entities in the LuisJSONModel object and applies some conditional logic before applying the value to static variables. Once the program has iterated over all the entities, the CreateSparqlQuery() is called and all the static variables are passed to it. You can see this in the image below (figure 8.4.6)

```

1 reference | leuan Walker, 11 days ago | 1 author, 2 changes
public static string ExtractLuisData(LuisJSONModel luisJson)
{
    int numberOfItems = 0;
    string genre = "";
    int year = 0;
    string exactDate = "";

    foreach (var i in luisJson.entities)
    {
        switch (i.type)
        {
            case "builtin.number":
                if (int.TryParse(i.resolution.value, out int number))
                {
                    if (number < 1000)
                    {
                        numberOfItems = number;
                    }
                }
                break;

            case "genre":
                genre = i.entity;
                break;

            case "builtin.datetime.date":
                if (DateTime.TryParse(i.entity, out DateTime exactDateTime))
                {
                    exactDate = exactDateTime.ToString();
                }
                else if (int.TryParse(i.entity, out int yearDateTime) && (i.entity.Length == 4))
                {
                    year = yearDateTime;
                }
                break;
        }
    }

    return CreateSparqlQuery(numberOfItems, genre, year, exactDate);
}

```

Figure 8.4.6 The ExtractLuisData() method

The CreateSparqlQuery() method returns the custom SPARQL query. It does this by checking the variables passed to it by the ExtractLuisData() method and applying some logic to create individual string variables. Then all these individual strings are combined using String.Format() method in C# to create the complete SPARQL query. You can see this in the image below (see figure 8.4.7).

```

1 reference | leuan Walker, 12 days ago | 1 author, 1 change
static string CreateSparqlQuery(int numberOfItems, string genre, int year, string exactDate)
{
    string limit = numberOfItems > 0 ? String.Format("LIMIT({0})", numberOfItems) : "";
    string genreMatch = !String.IsNullOrEmpty(genre.Trim()) ? String.Format("FILTER ( regex (str(?genre), '{0}', 'i'))", genre) : "";
    string dateMatch = "";

    if (exactDate.Equals(DateTime.Now) && year.Equals(0))
    {
        //Means that both haven't been assigned
    }
    else if (!String.IsNullOrEmpty(exactDate.Trim()))
    {
        dateMatch = String.Format("FILTER ( regex (str(?releaseDate), '{0}', 'i'))", exactDate);
    }
    else if (!year.Equals(0))
    {
        dateMatch = String.Format("FILTER ((?releaseDate >= '{0}-01-01'^^xsd:date) && (?releaseDate < '{0}-12-31'^^xsd:date))", year);
    }

    string queryPattern =
        "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> " +
        "PREFIX db: <http://dbpedia.org/ontology/> " +
        "PREFIX prop: <http://dbpedia.org/property/> " +
        "SELECT ?movieLink ?title ?genreLink ?genre ?releaseDate " +
        "WHERE {{ " +
        "    ?movieLink rdf:type db:Film; " +
        "    foaf:name ?title. " +
        "    OPTIONAL {{ ?movieLink prop:genre ?genreLink. " +
        "        ?genreLink rdfs:label ?genre. " +
        "        FILTER(lang(?genre) = 'en') }}. " +
        "    OPTIONAL {{ ?movieLink <http://dbpedia.org/ontology/releaseDate> ?releaseDate }}. " +
        "    {0} " +
        "    {1} " +
        "    FILTER(lang(?title) = 'en') " +
        "}} " +
        "ORDER BY DESC(?releaseDate)" +
        "{2}";

    return String.Format(queryPattern, genreMatch, dateMatch, limit);
}

```

Figure 8.4.7 The Create SparqlQuery() method

Finally, it was time to add this to the main ASP.NET web application, but before doing so I needed to change the CallLuis() method. Here is the new CallLuis() method (see figure 8.4.8).

```

//Method to query the Luis.ai
1 reference | leuan Walker, 10 days ago | 1 author, 1 change | 0 exceptions
public static async Task<LuisJSONModel> CallLuisAsync(string Query)
{
    LuisJSONModel Data = new LuisJSONModel();
    using (HttpClient client = new HttpClient())
    {
        string LUIS_Url = WebConfigurationManager.AppSettings["LUIS_Url"];
        string LUIS_Id = WebConfigurationManager.AppSettings["LUIS_Id"];
        string LUIS_Subscription_Key = WebConfigurationManager.AppSettings["LUIS_Subscription_Key"];
        string LUIS_Query = Uri.EscapeDataString(Query);

        string RequestUri = String.Format("{0}{1}?subscription-key={2}&verbose=true&q={3}", LUIS_Url, LUIS_Id, LUIS_Subscription_Key, LUIS_Query);
        Console.WriteLine(RequestUri);

        HttpResponseMessage msg = await client.GetAsync(RequestUri);

        if (msg.IsSuccessStatusCode)
        {
            var JsonDataResponse = await msg.Content.ReadAsStringAsync();
            Debug.WriteLine(JsonDataResponse);
            Data = JsonConvert.DeserializeObject<LuisJSONModel>(JsonDataResponse);
        }
    }

    return Data;
}

```

As you can see the method has changed considerably. This is because instead of simply reading a file and returning an object, it is now taking a string, creates a query URI and uses the HTTP endpoint for the Luis app that I created earlier in section 8.2.4. But the theory is the same, the method gets called and it returns LuisJSONModel object.

8.5 SPARQL search

Above in section 8.4, I discussed how I created a SPARQL query using the user input and the LUIS application. This section is going to be relatively short because I'll be discussing how to send a SPARQL query to DBpedia and what is returned.

To send the SPARQL query to DBpedia I created a simple method called QueryDbpedia(). When the method is called, the SPARQL query generated above (*section 8.4*) is passed as a parameter.

I then connected to the DBpedia SPARQL endpoint and sent off the SPARQL query. In return, I received a SparqlResultSet. Here is the method (*see figure 8.5.1*).

```
//Method to Query Dbpedia and return a Sparql Result set
2 references | Ieuan Walker, 12 days ago | 1 author, 1 change | 0 exceptions
public static SparqlResultSet QueryDbpedia(string query)
{
    //Define a remote endpoint
    //Use the DBPedia SPARQL endpoint with the default Graph set to DBPedia
    SparqlRemoteEndpoint endpoint = new SparqlRemoteEndpoint(new Uri("http://dbpedia.org/sparql"), "http://dbpedia.org");

    //Make a SELECT query against the Endpoint
    SparqlResultSet results = endpoint.QueryWithResultSet(query);

    return results;
}
```

Figure 8.5.1 The method that queries DBpedia (QueryDbpedia())

In section 8.6.2 below I will discuss how I processed the SparqlResultSet to the database.

8.6 Database – Entity framework code first approach

To create the database, I used Entity Framework code first approach. Microsoft (Microsoft, 2016) explains entity framework as an object-relational mapper that allows developers to work with relational data as objects.

I decided to create the database using the code first approach. What this means is that the table is created by a class object. Here is the database I created.

```
7 references | Ieuan Walker, 12 days ago | 1 author, 1 change
public class MovieUserSearch
{
    [Key]
    1 reference | Ieuan Walker, 12 days ago | 1 author, 1 change | 0 exceptions
    public int ID { get; set; }
    [Required]
    8 references | Ieuan Walker, 12 days ago | 1 author, 1 change | 0 exceptions
    public string SearchedFor { get; set; }
    2 references | Ieuan Walker, 12 days ago | 1 author, 1 change | 0 exceptions
    public string MovieLink { get; set; }
    2 references | Ieuan Walker, 12 days ago | 1 author, 1 change | 0 exceptions
    public string Title { get; set; }
    2 references | Ieuan Walker, 12 days ago | 1 author, 1 change | 0 exceptions
    public string GenreLink { get; set; }
    2 references | Ieuan Walker, 12 days ago | 1 author, 1 change | 0 exceptions
    public string Genre { get; set; }
    2 references | Ieuan Walker, 12 days ago | 1 author, 1 change | 0 exceptions
    public string ReleaseDate { get; set; }
    3 references | Ieuan Walker, 12 days ago | 1 author, 1 change | 0 exceptions
    public DateTime LastUpdated { get; set; } = DateTime.Now;
}
```

Figure 8.6.1 Database design code first approach

To initialize and create the database there are only two more steps. First I had to create a new class to hold the DbContext (see figure 8.6.2) and then add the DbContext to the web.config file to create a connection to a SQL server (see figure 8.6.3).

```
8 references | Ieuan Walker, 12 days ago | 1 author, 1 change
public class MovieDbContext : DbContext
{
    9 references | Ieuan Walker, 12 days ago | 1 author, 1 change | 0 exceptions
    public DbSet<MovieUserSearch> movieUserSearchTable { get; set; }
}
```

Figure 8.6.2 Creating the DbContext

```
<connectionStrings>
  <add name="MovieDbContext" connectionString="Data Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\movieSearch.mdf;Integrated Security=True" providerName="System.Data.SqlClient" />
</connectionStrings>
```

Figure 8.6.3 The connection string

Within the application, I used the database extensively to perform all sorts of operations such as checking if a query already exists, checking the age of the result, deleting old records and adding new results. To perform all these actions, I used a query language called LINQ. “Query (LINQ), is a component released within the .NET 3.5 Framework. It is one of the most powerful features of .NET 3.5. It serves the purpose of querying objects.” (Gupta, 2015). So, by utilising LINQ it offers a fast and effective way to work with the database.

For example, I used it to check if a search already exists (see figure 8.6.4) and I also used it to check when the queried data was last updated (see figure 8.6.5).

```
//Check if search exists
if (db.movieUserSearchTable.Any(o => o.SearchedFor.Equals(query)))
```

Figure 8.6.4 Search query

```
var queryDate = from s in db.movieUserSearchTable
                 where s.SearchedFor == query
                 select s;

var dateTimeNow = DateTime.Now;
var dateTimeOldest = dateTimeNow.AddMinutes(minutesOld);
var lastUpdated = queryDate.FirstOrDefault().LastUpdated;
```

Figure 8.6.5 Query last updated

8.6.1 Deleting from the database

One of the actions I needed to do was to delete data from the database if the data was too old. The reason for this is to ensure that the user has up to date information. Here is the code that handles the operation (see figure 8.6.1.1)

```

//Information in database is too old, need newer information
//Delete current information
var moviesSearch = from s in db.movieUserSearchTable select s;
moviesSearch = moviesSearch.Where(s => s.SearchedFor.Equals(query));
foreach (MovieUserSearch i in moviesSearch)
{
    db.movieUserSearchTable.Remove(db.movieUserSearchTable.Find(i.ID));
}
try
{
    db.SaveChanges();
}
catch (Exception e)
{
    Debug.WriteLine("ERROR: Removing items from database");
    Debug.WriteLine(e);
}
//Request new information
await gatherNewDataAsync(query);

```

Figure 8.6.1.1 Deleting from the database

As you can see it uses a LINQ query to gather all the results that need to be deleted. It then loops through them all and deletes them one by one. Once it has looped through all the results it then saves the changes to the database within a try catch block to stop the application from crashing if there was an error.

8.6.2 Adding to the database

To add data to the database, I used two methods.

The first method `loopValueToDatabase()`, takes a `SparqlReultSet`. The `SparqlResultSet` is the returned result from the DBpedia query. The method loops through each result parsing the information into the correct variable types and formats, before passing the variable to a new method called `AddToDatabase()`. The image below shows the actual method (see figure 8.6.2.1).

```

1 reference | leuan Walker, 10 days ago | 1 author, 4 changes | 0 exceptions
private void loopValuesToDatabase(string searchString, SparqlResultSet resultSet)
{
    foreach (SparqlResult result in resultSet)
    {
        string movieLink = result["movieLink"].ToString();
        string title = utilities.RemoveLast3Cahacters(result["title"].ToString());
        string genreLink = "";
        if (!(result["genreLink"] == null))
        {
            genreLink = result["genreLink"].ToString();
        }
        string genre = "";
        if (!(result["genre"] == null))
        {
            genre = utilities.RemoveLast3Cahacters(result["genre"].ToString());
        }
        string releaseDate = "";
        if (!(result["releaseDate"] == null))
        {
            releaseDate = utilities.DateCreator(result["releaseDate"].ToString());
        }
        AddToDatabase(searchString, movieLink, title, genreLink, genre, releaseDate);
    }
}

```

Figure 8.6.2.1

The loopValueToDatabase() method

The second method AddToDatabase(), does exactly what it says. It adds a single record to the database. It does it by creating a MovieUserSearch object (*see section 8.6*), adding it to the database and then saving the changes (*see figure 8.6.2.2*).

```
1 reference | leuan Walker, 12 days ago | 1 author, 1 change | 0 exceptions
private void AddToDatabase(string searchString, string movieLink, string tilte, string genreLink, string genre, string releaseDate)
{
    //Create movie search object
    MovieUserSearch movieSearch = new MovieUserSearch
    {
        SearchedFor = searchString,
        MovieLink = movieLink,
        Title = tilte,
        GenreLink = genreLink,
        Genre = genre,
        ReleaseDate = releaseDate
    };

    //Add the new object to the table
    db.MovieUserSearchTable.Add(movieSearch);

    //Submit change to database
    try
    {
        db.SaveChanges();
    }
    catch (Exception e)
    {
        Console.WriteLine("ERROR: Adding to database");
        Console.WriteLine(e);
    }
}
```

Figure 8.6.2.2

The AddToDatabase() method

8.7 Search Process

The above implementations are all small components of the overall program. To combine them all I have designed a simple method called searchAsync() inside the movieSearch class. This method is the first method to get called when a search is initiated. Here is the method (*see figure 8.7.1*) and this is the implementation of the design I discussed above in section 7.3.

```

2 references | Ieuan Walker, 12 days ago | 1 author, 4 changes
public class movieSearch
{
    private readonly MovieDBContext db = new MovieDBContext();
    private int minutesOld = -2;

    1 reference | Ieuan Walker, 13 days ago | 1 author, 1 change | 0 exceptions
    public async Task searchAsync(string query)
    {
        //Check if search exists
        if (db.movieUserSearchTable.Any(o => o.SearchedFor.Equals(query)))
        {
            var queryDate = from s in db.movieUserSearchTable
                            where s.SearchedFor == query
                            select s;

            var dateTimeNow = DateTime.Now;
            var dateTimeOldest = dateTimeNow.AddMinutes(minutesOld);
            var lastUpdated = queryDate.FirstOrDefault().LastUpdated;

            //check date
            if (lastUpdated <= dateTimeNow && lastUpdated >= dateTimeOldest)
            {
                //Information in the database is within the date period no need to get new information
            }
            else if (lastUpdated < dateTimeOldest)
            {
                //Information in database is too old, need newer information
                //Delete current information
                var moviesSearch = from s in db.movieUserSearchTable select s;
                moviesSearch = moviesSearch.Where(s => s.SearchedFor.Equals(query));
                foreach (MovieUserSearch i in moviesSearch)
                {
                    db.movieUserSearchTable.Remove(db.movieUserSearchTable.Find(i.ID));
                }
                try
                {
                    db.SaveChanges();
                }
                catch (Exception e)
                {
                    Debug.WriteLine("ERROR: Removing items from database");
                    Debug.WriteLine(e);
                }

                //Request new information
                await gatherNewDataAsync(query);
            }
            else if (lastUpdated > dateTimeNow)
            {
                Debug.WriteLine("ERROR: Database information is newer than possible");
            }
        }
        else
        {
            //Request new information
            await gatherNewDataAsync(query);
        }
    }
}

```

Figure 8.7.1 The searchAsync() method

The image above (*figure 8.7.1*) shows the process a search can take. It combines all the discussed aspects above to work together and make the process quick and efficient. I will not discuss the code above because I have explained the process previously in section 7.3.

9. Results and discussion

This is the final section of my dissertation where I will discuss all my findings and answer my research questions. As a quick reminder of my aims and objectives (see section 2), my main research question is *“Can semantic web be combined with AI, for example, Natural Language Processing to create a smart search engine?”*. But to answer this question I created two questions which were, ‘why would a semantic search engine be better than a traditional search engine (Google/ Bing)?’ and ‘can semantic web data be combined with a traditional relational database?’. I also devised four objectives to complete throughout the dissertation to answer my research question. Here are the objectives -

1. *“Investigate current search engines and explain how a semantic search engine will differ”*
2. *“Explain the benefits of semantic data”*
3. *“Investigate if semantic data can be combined with a traditional database”*
4. *“Develop a simple semantic web application using natural language processing”*

Note: I'd highly recommend watching the video demo of the web application before reading further. You can find the video in the submitted documents or at the following link – <https://youtu.be/luJhd5diDgg>

After watching the video, you will notice that I have achieved one of my objectives, *“Develop a simple semantic web application using natural language processing”*

In the following subsections, I will be answering all my questions and objectives, but before I answer these questions, I am going to discuss the limitations of my research.

9.1 Limitations

In this section, I am going to discuss the limitations of my research and show how someone could develop my research further.

Firstly, in my project, I used SQL server to store the returned results, and in section 4.2.2 I justified my decision, but someone could extend my research by exploring, comparing and implementing the other options such as MySQL, Oracle, SQL Server, etc.

The main language I used to create the web application was C# and in section 4.2.1 explain my decision. In the section, I discussed the different languages I could have used such as C++, JavaScript, C# and Java, but it ended in a close decision between C# and Java. It would be interesting to see if the project could be recreated in another language and if there would be any advantages of doing so.

Someone could also do some research into the reasoning systems, from what they are to how to use them.

And finally, I used LUIS.ai for the natural language processing and I chose this simply because it was developed by Microsoft (which lowers the chance of compatibility issues) and it had in-depth guides for use with C#. It would be interesting to see research into the

different natural language processors, to find out which is faster, easier to implement and which ones are better for different situations.

9.2 Current search engines VS a semantic search engine

Now I am going to answer one of my questions and address two of my objectives. The question I will be answering is 'why would a semantic search engine be better than a tradition search engine (Google/ Bing)?' and the objectives are *"Investigate current search engines and explain how a semantic search engine will differ"* and *"Explain the benefits of semantic data"*.

If you can remember back to section 1.1 I have already partially answered these questions. In the section, I discussed how traditional search engines (for example Google) works and how a semantic search engine would be better. As a quick re-cap, search engines indexes the web using bots and when a search is initiated it matches keywords from the input to the indexes. But the problem is that the same keywords can be used in different contexts and this is where the semantic web fits in. Semantic web adds understanding and meaning to the data making it easier to match results for a specific search. And in my research, I demonstrated how this can be further enhanced with the use of AI specifically NLP, because NLP adds more understanding and meaning to the users search meaning that there is a lot more understanding of all the data involved, allowing for more accurate results.

So, as you can see a system with more understanding of the data involved will benefit the users as it can easily identify the context of a search and return only the most meaningful results.

9.3 The combination of different technologies

In this subsection, I will be addressing one of the questions and one of the objectives. The question I will be answering is 'can semantic web data be combined with a traditional relational database?' and the objective is *"Investigate if semantic data can be combined with a traditional database"*.

As you can see from the video demonstration and the detailed implementation section (see section 8) it is possible to combine semantic web data with a traditional relational database.

The main reason I decided to implement a database into the project, is to demonstrate how multiple different technologies can be combined to give a better experience to the user. To demonstrate how the SQL database benefits the user I have created six tables to show the speeds of different searches (see section 'List of tables'). The tables have 3 columns the first tilted 'search string' is simply the search I conducted on the site. The second column labelled 'update status' shows whether the search 'needs updating' or the results are 'already updated'. And the final column labelled 'time', shows how long the search took to process in milliseconds.

Note: To record the time the method took to complete, I used a system method called `stopwatch()`. And according to Microsoft, it is used “*to accurately measure elapsed time*” (Microsoft, n.d.). You can see how I implemented this function in figure 9.3.1.

```
Stopwatch timer = Stopwatch.StartNew();  
await UserSearch.searchAsync(searchQuery);  
timer.Stop();  
Debug.WriteLine(timer.ElapsedMilliseconds);
```

Figure 9.3.1 Implementation of the `stopwatch()` method

As you can see from the tables, with the data ‘already updated’ (table 2,4,6) the time ranges from 61 to 97ms which is extremely fast from the user point of view, as at the slowest it is less than 0.1 of a second. But when you look at the searches that ‘need updating’ (tables 1, 3, 5) the average time ranges from 1893 to 7811ms. As you can see it takes much longer than the results already stored in the database, and this is because it is processing a lot more. As you can see from section 7.3, when updating results in the database there are several steps to go through, these are –

1. Delete old results from the database
2. Call Luis and extract the data
3. Create SPARQL query
4. Send SPARQL query to DBpedia
5. Add results to database

You may also notice there is a big range difference, of nearly 6s, and compared to the ‘already updated’ with less than 0.1 of a second, it will be noticeable to the user. There are a lot of reason why the searches that ‘need updating’ can range so drastically, here is a list why –

- The more complex a SPARQL query is, the longer it takes for DBpedia to return a result
- If DBpedia finds a lot of results, it will take longer to return a result
- And finally, the more results returned from DBpedia the longer it will take to process and add to the database

I have created a chart (see figure 9.3.2), as you can see the chart the data that is ‘Already Updated’ consistently fast, and the ‘Need Updating’ varies a lot.

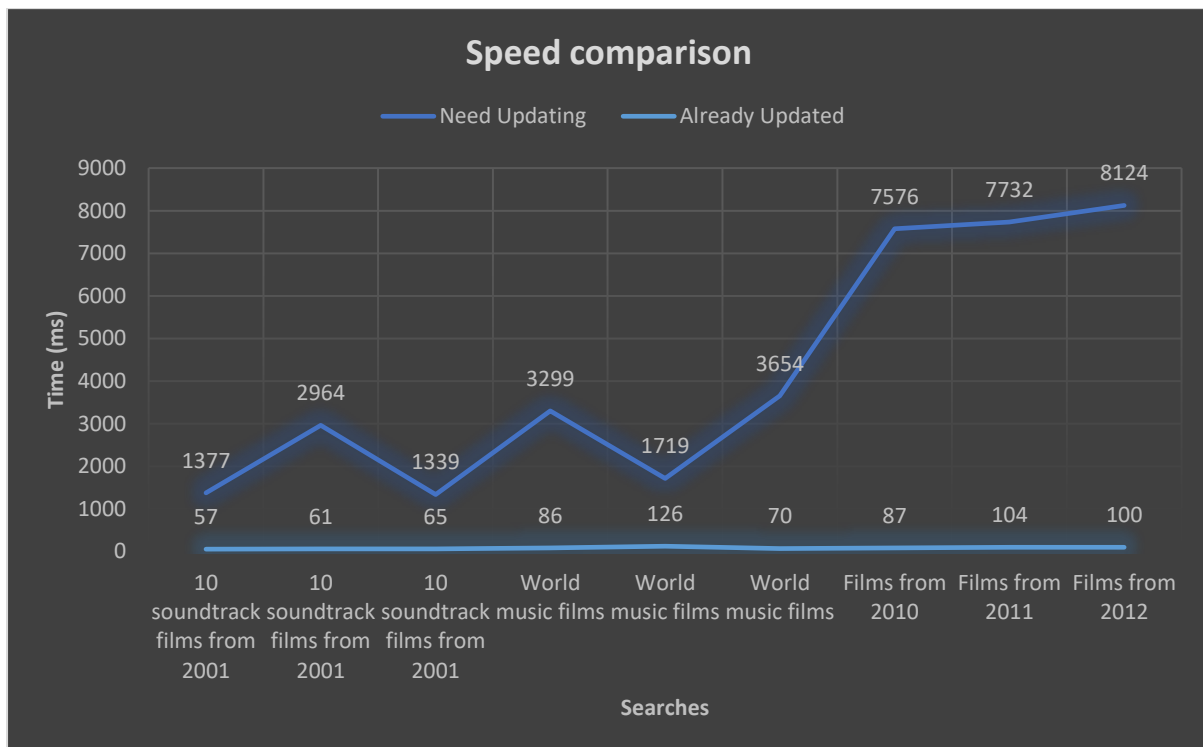


Figure 9.3.2 Chart showing speed comparisons

The above results not only answer the question ‘*can semantic web data be combined with a traditional relational database?*’ but also shows the huge benefit of doing so, by lowering the wait time from 6 seconds to 0.1 of a second.

9.4 Conclusion

So, it is now time to conclude my dissertation. I will now address my main research question which is “*Can semantic web be combined with AI, for example, Natural Language Processing to create a smart search engine?*”.

As you can hopefully see my project was a success, I successfully combined the semantic web with natural language processing. The combination of these two semantic technologies created a unique experience by allowing a user to simply ask for different pieces of information (genre, dataTime and number of results) and use the semantic web to retrieve exactly what the user wanted. The is powerful and could easily be utilised in a lot of different sectors in the future.

After all the research I have conducted, I believe that the semantic web will be an important milestone in interconnecting data across the web. The power of the semantic web is vast. It is already advancing the power of AI and scientist are adding more and more data to the semantic web which speeding up research time. So the possibilities for the semantic web is endless.

References

Alghazawi, M., 2016. *Bot Framework in-depth: How to understand the message using Language Understanding Intelligent Service – LUIS*. [Online]

Available at: <http://streamcode.io/luis-in-depth/>

[Accessed 02 April 2017].

Allenmang, D. & Hendler, J., 2011. *Semantic Web for the Working Onologist - Effective Modeling in RDFS and OWL*. 2nd ed. Waltham: Morgan Kaufmann.

asp.net, n.d. *Get building*. [Online]

Available at: <https://www.asp.net>

[Accessed 20 February 2017].

BBC, n.d. *How do search engines work?*. [Online]

Available at: <http://www.bbc.co.uk/guides/ztbjq6f>

[Accessed 04 February 2017].

Berners-Lee, T., 2006. *Linked Data - Design Issues*. [Online]

Available at: <https://www.w3.org/DesignIssues/LinkedData.html>

[Accessed 14 April 2017].

Berners-Lee, T., 2009. *The next web*. [Online]

Available at: https://www.ted.com/talks/tim_berners_lee_on_the_next_web

[Accessed 4 February 2017].

Berners-Lee, T., Bizer, C. & Heath, T., n.d. *Linked Data - The Story So Far*. [Online]

Available at: <http://tomheath.com/papers/bizer-heath-berniers-lee-ijswis-linked-data.pdf>

[Accessed 09 April 2017].

Berners-Lee, T., Hendler, J. & Lassila, O., 2001. *Scientific American - The Semantic web*. [Online]

Available at: [https://www-](https://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American_%20Feature%20Article_%20The%20Semantic%20Web_%20May%202001.pdf)

[sop.inria.fr/acacia/cours/essi2006/Scientific%20American %20Feature%20Article %20The %20Semantic%20Web %20May%202001.pdf](https://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American_%20Feature%20Article_%20The%20Semantic%20Web_%20May%202001.pdf)

[Accessed 3 January 2017].

Business Dictionary, n.d. *Relational Database*. [Online]

Available at: <http://www.businessdictionary.com/definition/relational-database.html>

[Accessed 01 April 2017].

Cambridge Semantics, n.d. *Introduction to the Semantic Web*. [Online]

Available at: <http://www.cambridgesemantics.com/semantic-university/introduction-semantic-web>

[Accessed 11 April 2017].

- Cambridge Semantics, n.d. *RDFS vs. OWL*. [Online]
Available at: <http://www.cambridgesemantics.com/semantic-university/rdfs-vs-owl>
[Accessed 17 April 2017].
- Cambridge Semantics, n.d. *SPARQL 101*. [Online]
Available at: <http://www.cambridgesemantics.com/semantic-university/sparql-101>
[Accessed 12 April 2017].
- Fan, W., Wang, X. & Wu, Y., 2013. Incremental graph pattern matching. *ACM Transactions on Database Systems (TODS)*, 38(3).
- Futrell, R. & Gruber, T., 2016. *Exemplar-based natural language processing*. United States, Patent No. 9,430,463.
- git-scm, n.d. *Getting Started - About Version Control*. [Online]
Available at: <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>
[Accessed 01 April 2017].
- Gupta, S., 2015. *What is LINQ?*. [Online]
Available at: <http://www.c-sharpcorner.com/blogs/what-is-linq1>
[Accessed 04 April 2017].
- Heath, T. & Bizer, C., 2011. *Linked Data: Evolving the Web into a Global Data Space*. Berlin: Morgan & Claypool.
- Hebeler, J., Fisher, M., Blace, R. & Perez-Lopez, A., 2009. *Semantic Web Programming*. Chichester: Wiley.
- Idehen, K. U., 2016. *What is DBpedia, and why is it important?*. [Online]
Available at: <https://medium.com/openlink-software-blog/what-is-dbpedias-and-why-is-it-important-d306b5324f90>
[Accessed 02 April 2017].
- Jarrar, M., 2013. *Jarrar: RDFS (RDF Schema)*. [Online]
Available at: <https://www.slideshare.net/MustafaJarrarEdu/jarrarlecture-notesrdfs2>
[Accessed 18 April 2017].
- Jurafsky, D. & Martin, J. H., 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. illustrated ed. s.l.:Prentice Hall.
- Leinster, M., 1946. *A logic named Joe*. United States of America: Street & Smith.
- Maynard, D., Bontcheva, K. & Augenstein, I., 2016. Natural Language Processing for the Semantic Web. *Synthesis Lectures on the Semantic Web: Theory and Technology*, p. 194.

- McLeod, S., 2088. *Qualitative vs. Quantitative*. [Online]
Available at: <https://www.simplypsychology.org/qualitative-quantitative.html>
[Accessed 02 April 2017].
- Meehan, T., 2014. *Introduction to linked data*. [Online]
Available at:
<http://discovery.ucl.ac.uk/1449458/2/Intro%20to%20linked%20data%20C&I174final.pdf>
[Accessed 25 March 2017].
- Microsoft, 2016. *Introduction to Entity Framework*. [Online]
Available at: [https://msdn.microsoft.com/en-us/library/aa937723\(v=vs.113\).aspx](https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx)
[Accessed 04 April 2017].
- Microsoft, n.d. *Garbage Collection*. [Online]
Available at: [https://msdn.microsoft.com/en-us/library/0xy59wtx\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/0xy59wtx(v=vs.110).aspx)
[Accessed 20 March 2017].
- Microsoft, n.d. *Stopwatch Class*. [Online]
Available at: [https://msdn.microsoft.com/en-us/library/system.diagnostics.stopwatch\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.diagnostics.stopwatch(v=vs.110).aspx)
[Accessed 16 April 2017].
- Mircosoft, n.d. *Connected Services Framework*. [Online]
Available at: <https://msdn.microsoft.com/en-us/library/aa303673.aspx>
[Accessed 08 April 2017].
- Outlaw, R., n.d. *Git*. [Online]
Available at: <https://www.visualstudio.com/learn/what-is-version-control/>
[Accessed 2017 April 01].
- Powers, S., 2003. *Practical RDF*. Sebastopol: O'Reilly.
- Saunders, M., Lewis, P. & Adrian, T., 2007. *Research Methods for Business Students*. illustrated ed. s.l.:Pearson Education, 2007.
- Scrum Guides, n.d. *Scrum Guide*. [Online]
Available at: <http://www.scrumguides.org>
[Accessed 20 February 2017].
- Sequeda, J., 2011. *Introduction to: SPARQL*. [Online]
Available at: <http://www.dataversity.net/introduction-to-sparql/>
[Accessed 11 April 2017].
- Shiotsu, Y., n.d. *C# vs. Java: Which General Purpose Programming Language Is Best for You?*. [Online]
Available at: <https://www.upwork.com/hiring/development/c-vs-java/>
[Accessed 29 March 2017].

techopedia, n.d. *Console Application*. [Online]
Available at: <https://www.techopedia.com/definition/25593/console-application-c>
[Accessed 20 March 2017].

The Windows Club, n.d. *Difference between SQL and MySQL*. [Online]
Available at: <http://www.thewindowsclub.com/difference-sql-mysql>
[Accessed 01 April 2017].

Tutorials Point, n.d. *MVC Framework - Introduction*. [Online]
Available at:
https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
[Accessed 20 February 2017].

w3, 2009. *OWL*. [Online]
Available at: <https://www.w3.org/OWL/>
[Accessed 17 April 2017].

W3C, 2005. *SPARQL Query Language for RDF*. [Online]
Available at: <https://www.w3.org/TR/2005/WD-rdf-sparql-query-20050217/#GraphPatternMatching>
[Accessed 12 April 2017].

W3C, 2006. *Naming and Addressing: URIs, URLs, ...* [Online]
Available at: <https://www.w3.org/Addressing/>
[Accessed 08 April 2017].

w3c, 2013. *SPARQL 1.1 Query Language*. [Online]
Available at: <https://www.w3.org/TR/sparql11-query/#specifyingDataset>
[Accessed 13 April 2017].

w3c, n.d. *Linked data*. [Online]
Available at: <https://www.w3.org/standards/semanticweb/data>
[Accessed 25 March 2017].

w3, n.d. *What is HyperText*. [Online]
Available at: <https://www.w3.org/WhatIs.html>
[Accessed 5 January 2017].

w3schools, n.d. *XML RDF*. [Online]
Available at: https://www.w3schools.com/xml/xml_rdf.asp
[Accessed 09 April 2017].

Appendix

Appendix 1 – Ethics Approval Form

ETHICS APPROVAL NUMBER: **2016D0188**

When undertaking a research or enterprise project, Cardiff Met staff and students are obliged to complete this form in order that the ethics implications of that project may be considered.

If the project requires ethics approval from an external agency (e.g., NHS), you will not need to seek additional ethics approval from Cardiff Met. You should, however, complete Part One of this form and attach a copy of your ethics letter(s) of approval in order that your School has a record of the project.

The document ***Ethics application guidance notes*** will help you complete this form. It is available from the [Cardiff Met website](#). The School or Unit in which you are based may also have produced some guidance documents, please consult your supervisor or School Ethics Coordinator.

Once you have completed the form, sign the declaration and forward to the appropriate person(s) in your School or Unit.

PLEASE NOTE:

Participant recruitment or data collection MUST NOT commence until ethics approval has been obtained.

PART ONE

Name of applicant:	Ieuan Walker
Supervisor (if student project):	Dr Ambikesh Jayal
School / Unit:	Cardiff Metropolitan University
Student number (if applicable):	ST20057645
Programme enrolled on (if applicable):	Software Engineering – BSc (Hons)
Project Title:	A comparison of between semantic web technologies and traditional databases.
Expected start date of data collection:	N/A (No primary data collection)
Approximate duration of data collection:	N/A (No primary data collection)
Funding Body (if applicable):	N/A
Other researcher(s) working on the project:	N/A
Will the study involve NHS patients or staff?	No

Will the study involve human samples and/or human cell lines?	No
---	----

Does your project fall entirely within one of the following categories:	
Paper based, involving only documents in the public domain	Yes
Laboratory based, not involving human participants or human samples	No

Practice based not involving human participants (eg curatorial, practice audit)	No
Compulsory projects in professional practice (eg Initial Teacher Education)	No
A project for which external approval has been obtained (e.g., NHS)	No
If you have answered YES to any of these questions, expand on your answer in the non-technical summary. No further information regarding your project is required. If you have answered NO to all of these questions, you must complete Part 2 of this form	

<p>In no more than 150 words, give a non-technical summary of the project</p> <p>I confirm that this research does not involve any primary data collection.</p> <p>My topic is semantic web; the semantic web is the idea of creating a standard representation of linked data to allow machine processing on a global scale. As part of my research, I will be describing the different technologies used in semantic web.</p> <p>This project will show whether semantic search engine and database is more efficient than the standard technologies.</p> <p>So, my project I will be creating a simple semantic search engine that use's inference and comparing the efficiency and speed against a standard relational database.</p>

<p>DECLARATION:</p> <p>I confirm that this project conforms with the Cardiff Met Research Governance Framework</p> <p>I confirm that I will abide by the Cardiff Met requirements regarding confidentiality and anonymity when conducting this project.</p> <p>STUDENTS: I confirm that I will not disclose any information about this project without the prior approval of my supervisor.</p>	
Signature of the applicant: Ieuan Walker (st20057645)	Date: 10/11/2016
FOR STUDENT PROJECTS ONLY	
Name of supervisor: Dr Ambikesh Jayal	Date:
Signature of supervisor:	

Research Ethics Committee use only	
Decision reached:	Project approved Project approved in principle Decision deferred Project not approved Project rejected
Project reference number: Click here to enter text.	
Name: Click here to enter text.	Date: Click here to enter a date.
Signature:	
Details of any conditions upon which approval is dependant: Click here to enter text.	

PART TWO

A RESEARCH DESIGN	
A1 Will you be using an approved protocol in your project?	Choose an item.
A2 If yes, please state the name and code of the approved protocol to be used ¹	
Click here to enter text.	
A3 Describe the research design to be used in your project	
In this section, include details (as appropriate) of: - research method(s); - sample and sampling; - recruitment of participants; - analytical techniques If your project does involve the use of an approved protocol, much less detail will be required but you should indicate which areas of the project are covered by the protocol.	
A4 Will the project involve deceptive or covert research?	Choose an item.
A5 If yes, give a rationale for the use of deceptive or covert research	
Click here to enter text.	
A6 Will the project have security sensitive implications?	Choose an item.
A7 If yes, please explain what they are and the measures that are proposed to address them	
Click here to enter text.	

B PREVIOUS EXPERIENCE

¹ An Approved Protocol is one which has been approved by Cardiff Met to be used under supervision of designated members of staff; a list of approved protocols can be found on the Cardiff Met website [here](#)

B1 What previous experience of research involving human participants relevant to this project do you have?
--

Click here to enter text.

B2 Student project only

What previous experience of research involving human participants relevant to this project does your supervisor have?

Click here to enter text.

C POTENTIAL RISKS

C1 What potential risks do you foresee?

Include details of risks to the participants, the researcher and the project as a whole.
--

C2 How will you deal with the potential risks?
--

Click here to enter text.

When submitting your application you **MUST** attach a copy of the following:

- All information sheets
- Consent/assent form(s)

An exemplar information sheet and participant consent form are available from the Research section of the Cardiff Met website.