The SPARQL Usage for Mapping Maintenance and Reuse Methodology

A thesis submitted to the
University of Dublin, Trinity College
in fulfilment of the requirements of the degree of
Doctor of Philosophy

Alan Meehan
Knowledge and Data Engineering Group (KDEG)
School of Computer Science and Statistics
Trinity College, Dublin
Ireland

Supervised by Prof. Declan O’Sullivan
Co-supervised by Dr. Rob Brennan

2017
Declaration

I, the undersigned, declare that this work has not been previously submitted as an exercise for a degree at this or any other University, and that, unless otherwise stated, it is entirely my own work.

Signature _________________________  Date _________________________

Alan Meehan
Permission to lend or copy

I, the undersigned, agree that the Trinity College Library may lend or copy this thesis upon request.

Signature ___________________________ Date ___________________

Alan Meehan
Acknowledgements

I would like to thank my supervisors, Declan O’Sullivan and Rob Brennan, for all of their support and guidance throughout this Ph.D. I could not have done it without them.

This research is supported by Science Foundation Ireland through the CNGL Program (Grant 12/CE/I2267) in the ADAPT Centre (www.adaptcentre.ie) at Trinity College Dublin.
Abstract

This thesis presents the SPARQL Usage for Mapping Maintenance and Reuse (SUMMR) methodology, which is for the support of performing maintenance and reuse of Linked Data mappings.

The providers of Linked Data datasets have made a great effort in recent years to publish more Linked Data on the web. These datasets cover a wide range of knowledge domains. For some datasets, their knowledge domains overlap. Due to the decentralised nature of their development, ownership and management they use heterogeneous vocabularies. This leads to diversity of vocabulary term representations and data instance representations for overlapping terms and instances. Mappings can be used to overcome these heterogeneities, but the creation of new mappings can be a difficult and time consuming task. Therefore, reusing existing mappings, to aid in the creation of new mappings becomes appealing to dataset maintainers. In addition datasets can change over time. These changes can effect existing mappings that reference the changed dataset, causing the mappings to become invalid and no longer produce correct results. Mapping maintenance is concerned with the discovery and repair of invalid mappings and this is also a difficult and time consuming task.

Through a systematic literature review, modelling and evaluation, the SUMMR methodology unifies both mapping maintenance and mapping reuse, through identifying use cases from both and combining them together, along with specific tasks, into a common methodology. SUMMR provides standard SPARQL query templates to perform maintenance and reuse use cases and tasks over RDF-based mapping representations and datasets. SUMMR provides a specialised mapping representation – named the SPARQL Centric Mapping Representation (SCMR). SCMR is designed to represent two categories of ontology mappings (i) those used to transform data described from one Linked Data vocabulary into another and (ii) interlinks between semantically similar instances in Linked Data datasets. The SCMR represents these categories of mappings with sufficient detail to maximise support for SUMMR templates. However, SUMMR can be used with alternative RDF-based mapping representations.

The SUMMR methodology has been evaluated through three lab-based experiments and a case study which involved maintenance of interlink category mappings in the DBpedia dataset. The lab-based experiments provide evidence that SUMMR templates can perform all mapping maintenance and reuse tasks over mappings represented in
SCMR. An additional lab-based experiment provides evidence the SCMR is sufficiently expressive for representing ontology mappings which are concerned with the transformation of data from one vocabulary to another. The case study evaluation was performed to evaluate the usefulness of SUMMR in a real world Linked Data dataset management situation. SUMMR was applied to the interlink management process of the DBpedia dataset, through an open-source software tool - named the SUMMR Interlink Validation Tool, to provide SUMMR-based invalid interlink category mapping detection. The SUMMR Interlink Validation Tool was used to validate 1,679,634 interlink category mappings that were to be published in the v.2015-10 DBpedia dataset release and discovered that 53,418 of these interlink category mappings were invalid.

The research in this thesis has yielded one major and two minor contributions. The major contribution is the design, development and evaluation of the SPARQL Usage for Mapping Maintenance and Reuse (SUMMR) methodology, for the support of performing mapping maintenance and reuse of vocabulary transformation and interlink category mappings. The minor contributions are the SPARQL Centric Mapping Representation (SCMR) – developed as part of SUMMR, and the SUMMR Interlink Validation Tool which is an open-source software tool which implements SUMMR-based invalid interlink category mapping detection.
Table of Contents

Declaration ........................................................................................................................................ ii
Permission to lend or copy ............................................................................................................... iii
Acknowledgements ....................................................................................................................... iv
Abstract .......................................................................................................................................... v
Table of Contents .......................................................................................................................... vii
List of Figures ................................................................................................................................... xii
List of Tables ...................................................................................................................................... xiv
Abbreviations .................................................................................................................................... xvi

1. Introduction .................................................................................................................................. 1
   1.1. Motivation ............................................................................................................................ 1
   1.2. Research Question ............................................................................................................... 4
       1.2.1. Definitions .................................................................................................................... 4
       1.2.2. Research Objectives .................................................................................................... 6
       1.2.3. Thesis Contributions .................................................................................................... 7
       1.2.4. Assumptions and Constraints ....................................................................................... 10
   1.3. Research Methodology ......................................................................................................... 11
   1.4. Evaluation Strategy ............................................................................................................. 12
   1.5. Thesis Overview .................................................................................................................. 15

2. Background ................................................................................................................................... 17
   2.1. Ontology Matching and Mapping ......................................................................................... 17
       2.1.1. The Matching Process .................................................................................................. 17
       2.1.2. The Mapping Process .................................................................................................. 18
   2.2. Linked Data Mappings ......................................................................................................... 19
       2.2.1. Uplift and Lowering Category of Mappings ................................................................. 19
       2.2.2. Vocabulary Category of Mappings .............................................................................. 20
       2.2.3. Interlink Category of Mappings ................................................................................... 20
   2.3. Chapter Summary ................................................................................................................ 21

3. State of the Art ............................................................................................................................. 22
   3.1. Introduction ........................................................................................................................ 22
   3.2. Existing Approaches for Mapping Maintenance and Reuse .......................................... 22
4.1.2. Refined Use Case 2: Discover Information from Existing Mappings to Aid in the Creation of New Mappings ................................................................. 51
4.1.3. Refined Use Case 3: Discover Mapping Paths to Aid in the Creation of New Mappings ........................................................................................................ 53
4.1.4. Refined Use Case 4: Discover Back Links to Aid in the Creation of New Mappings ........................................................................................................ 55
4.1.5. Refined Use Case 5: Discover Invalid Mappings due to Changes in Datasets a Mapping References .............................................................................. 56
4.1.6. Refined Use Case 6: Repair of an Invalid Mapping ..................................... 58
4.2. Chapter Summary .......................................................................................... 60

5. Design of the SUMMR Methodology ............................................................. 61

5.1. Introduction to the SUMMR Methodology ..................................................... 61
5.1.1. Brief Overview of SUMMR ........................................................................ 62
5.2. Elements of SUMMRs Design ..................................................................... 63
5.2.1. SPARQL Centric Mapping Representation (SCMR) .................................. 63
5.2.2. SUMMR Templates .................................................................................. 67
5.2.2.1. Maintenance and Reuse Task Templates ............................................... 68
5.2.2.2. Maintenance and Reuse Use Case Templates ........................................ 73
5.2.3. SUMMR Templates Summary ................................................................ 76
5.3. Operation of the SUMMR Methodology ...................................................... 76
5.3.1. Operation of SUMMR in a Mapping Lifecycle ......................................... 78
5.4. SUMMR Implementation ............................................................................ 80
5.4.1. SUMMR Interlink Validation Tool ............................................................ 80
5.4.2. Potential SUMMR Tools ......................................................................... 83
5.4.3. SUMMR Implementation Summary ....................................................... 83
5.5. Reflections upon the Design of SUMMR .................................................... 84
5.5.1. Features and Limitations ......................................................................... 84
5.5.2. Specialising SUMMR for Mapping Maintenance and Reuse for Other Domains 85
5.6. Chapter Summary ....................................................................................... 87

6. Evaluation ....................................................................................................... 88

6.1. Introduction .................................................................................................. 88
6.2. Experiment 1: Evaluation of the SPARQL Centric Mapping Representation ................................................................. 91
6.2.1. Introduction .............................................................................................. 91
6.5.6. Conclusion ........................................................................................................... 128
6.6. Chapter Summary ................................................................................................. 129
7. Conclusion ................................................................................................................ 131

7.1. Evaluation of Achievement of Research Objectives ............................................ 131

7.1.1. Objective 1: Review Existing Approaches for Mapping Maintenance and Reuse .......................................................... 131

7.1.2. Objective 2: Design a Query-based Methodology for Mapping Maintenance and Reuse ......................................................... 132

7.1.3. Objective 3: Implement the Query-based Methodology .................................... 133

7.1.4. Objective 4: Evaluate the Ability of the Query-based methodology to Support Performing Mapping Maintenance and Reuse ........................................... 133

7.1.5. Objective 5: Evaluate the Usefulness of the Query-based Methodology in a Real World Linked Data Situation .................... 135

7.2. Contributions ......................................................................................................... 135

7.3. Further Work ......................................................................................................... 137

7.4. Final Remarks ...................................................................................................... 138

References ...................................................................................................................... 139

Appendix A. SUMMR Use Case Template Examples .................................................... 147

Appendix B. DVD Contents .......................................................................................... 150

Appendix C. DBpedia Dataset v.2015-10 Release Email ................................................ 151
List of Figures

Figure 2-1. Depiction of the matching process. ................................................................. 18
Figure 2-2. Example of data uplifted from XML to RDF and lowered back to XML [Bi12]. ............................................................................................................................ 19
Figure 2-3. Example of data described according to the DBpedia vocabulary, transformed to be described according to the Linked Movie Database vocabulary. ............ 20
Figure 2-4. Examples of four interlinks from the DBpedia dataset to the Linked Movie Database, FreeBase, OpenCyc and YAGO datasets. ......................................................... 21
Figure 3-1. Classes and properties of the DSNotify Eventset Vocabulary [Po11]. ........... 25
Figure 3-2. Core Elements of the Change History Log [Kh15] ........................................... 26
Figure 3-3. LinkLion mapping representing an interlink category mapping between the DBpedia and Linked Movie Database datasets. ......................................................... 38
Figure 3-4. BioPortal Mapping representing an interlink category mapping between two biomedical databases in the BioPortal repository. ................................................ 39
Figure 3-5. R2R mapping for a movie runtime vocabulary resource between the DBpedia and Linked Movie Database datasets. ................................................................. 41
Figure 4-1. UML diagram for the refined use case of retrieving specific mappings for sharing ....................................................................................................................... 51
Figure 4-2. UML diagram for the refined use case of discovering information from existing mappings to aid in the creation of new mappings. ................................. 52
Figure 4-3. UML diagram for the refined use case of discovering mapping paths to aid in the creation of new mappings. ................................................................. 54
Figure 4-4. UML diagram for the refined use case of discovering back links to aid in the creation of new mappings. ................................................................. 56
Figure 4-5. UML diagram for the refined use case of discovering invalid mappings due to changes in datasets a mapping references. ........................................ 57
Figure 4-6. UML diagram for the refined use case of the repair of an invalid mapping. 59
Figure 5-1. Overview of the SUMMR Methodology .................................................................. 63
Figure 5-2. Example of SPARQL CONSTRUCT Query Encoded in RDF via the SPIN SPARQL Syntax. ................................................................................................. 64
Figure 5-3. Example of a SCMR Vocabulary Transformation Category Mapping. .......... 66
Figure 5-4. Example of an SCMR Interlink Category Mapping ........................................... 66
Figure 5-5. SUMMR Task Template 1 Specialised for SCMR ........................................... 70
Figure 5-6. SUMMR Task Template 2 Specialised for SCMR ........................................... 71
List of Tables

Table 3-1. Breakdown of existing approaches as to whether they support performing maintenance and reuse over mappings or alignments, mappings or alignments derived from complex correspondences and whether they are domain independent or not. ........36
Table 3-2. Six mapping maintenance and reuse use cases. ........................................43
Table 3-3. Five mapping maintenance and reuse tasks. ............................................46
Table 3-4. Mapping maintenance and reuse use cases and tasks supported by the existing approaches reviewed from the state of the art. .................................47
Table 3-5. Characteristics of Existing Mapping Representations Reviewed. ..................48
Table 5-1. SUMMR Mapping Maintenance and Reuse Task Templates .................68
Table 5-2. SUMMR Use Case Templates ...............................................................73
Table 6-1. Mapping Maintenance and Reuse Tasks. ..............................................90
Table 6-2. Number of identical and non-identical execution results of mappings from the R2R and SP-CON vocabulary mapping datasets. .............................97
Table 6-3. Number of mappings able and not able to be encoded in RDF via the SPIN SPARQL syntax. .................................................................98
Table 6-4. Average number of triples per mapping broken down by correspondence pattern. ...........................................................................................................98
Table 6-5. Mapping retrieval use cases from Thomas et al [Th11]. .........................103
Table 6-6. Experiment 2 mapping retrieval use cases. ..........................................105
Table 6-7. Concrete experiment 2 mapping retrieval use cases. .............................106
Table 6-8. Recall (R) and Precision (P) results of the bespoke queries and SUMMR templates performed over mappings represented in SCMR and the R2R Mapping Language for the 12 concrete mapping retrieval use cases. .........................108
Table 6-9. Mean execution time of the bespoke queries and templates performed over mappings represented in SCMR and the R2R Mapping Language for the 12 concrete use cases. .................................................................109
Table 6-10. Recall and Precision results of SUMMR Use Case Template 5 specialised for SCMR to discover invalid vocabulary transformation category mappings and invalid interlink category mappings and the invalid mappings ID numbers. ...............118
Table 6-11. Repaired vocabulary transformation category mappings execution results, by mapping ID number, compared to the mapping execution results gold standard dataset, categorised as either identical or not identical. ........................................118
Table 6.12. Repaired vocabulary transformation category mappings execution results, by mapping ID number, compared to the mapping execution results gold standard dataset, categorised as either identical or not identical. ..................................................119

Table 6-13. Detailed breakdown of the interlink sets validated by SUMMR QT5, categorized by external dataset name and indicating the number of interlinks per set, whether the interlink set was validated against an external dataset and the number and percentage of invalid interlink category mappings discovered per set, when the URI-to-IRI mapping were not considered and when they were considered. ........................................127

Table 6.14. Breakdown for the interlink sets that were externally validated of the number of interlink category mappings that were invalidated by changes in the DBpedia dataset and changes in the external dataset. .................................................................128

Table 7-1. SUMMR Use Case Templates used in parts 1 and 2 of experiment 3. .......135
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV</td>
<td>Comma Separated Values</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>SCMR</td>
<td>SPARQL Centric Mapping Representation</td>
</tr>
<tr>
<td>SPARQL</td>
<td>SPARQL Protocol and RDF Query Language</td>
</tr>
<tr>
<td>SUMMR</td>
<td>SPARQL Usage for Mapping Maintenance and Reuse</td>
</tr>
<tr>
<td>TSV</td>
<td>Tab Separated Values</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
1. Introduction

1.1. Motivation

The Linked Data principles [Bi09] were set out by Tim Berners-Lee as guidance for data providers when creating Linked Data datasets and publishing them on the web. A Linked Data dataset is structured information published with web technology, represented in the Resource Description Framework (RDF), linked to other datasets and accessible through HTTP [Bi08]. Based on the Linked Data principles, data providers have published over 1000 datasets on the web [Je11, Sc14]. One example is the work by Debruyne et al. [De16] which aims to provide Ireland’s national geospatial data as Linked Data. A well-known project called the Linking Open Data [Bi09] project has a goal to publish open (freely available data) datasets according to the Linked Data principles. These open datasets are now collectively known as the Linked Open Data cloud. According to two surveys of the Linked Open Data cloud [Je11, Sc14], the number of datasets in the cloud has increased from 295 in 2011 to 1014 in 2014. This increase in the number of datasets in the Linked Open Data cloud shows that data providers are making a concerted effort to grow the cloud and it is likely to keep growing given the current trends.

Datasets in the Linked Open Data cloud cover a wide range of knowledge domains and in some cases, their knowledge domains overlap. For example, the DBpedia [Le15], FreeBase [Bo09], GeoNames [Ah13] and LinkedGeoData [St12] datasets all contain geographical information and DBpedia, FreeBase and LinkedMovieDatabase [Ha09] also contain information about movies. Due to the de-centralised nature of their development, ownership and management, these datasets use heterogeneous vocabulary term representations\(^1\) and instance representations\(^2\) to model their data. These heterogeneities can cause interoperability issues between the datasets [Bi10] which can lead to implementation problems for supporting tasks such as information sharing, data integration and query answering [Eu07]. Ontology mappings\(^3\) [Eu13] can be created

---

1. Different vocabularies can have different representations of the same class or property. For example the DBpedia and the Linked Movie Database datasets both contain separate representations of the class “Film”.

2. Different datasets can have different representations of the same instance. For example, the DBpedia and the GeoNames datasets contain representations of the concept of the country of Ireland that differ in terms of RDF graph structure, vocabularies used, completeness and so on.

3. A mapping, as defined by Euzenat and Shvaiko [Eu07] is “…the oriented, or directed, version of an alignment: it maps entities [or resources] of one ontology [or dataset] to at most one entity [or resource] of another [dataset]”.

1
between the datasets to overcome data heterogeneity and hence improve interoperability. Mappings can be used to transform data from the vocabularies of multiple datasets into a common vocabulary\(^4\) or through linking related instances across the datasets\(^5\) (no. 4 of the Linked Data principles [Bi09]). Transformation of data from non-RDF data formats into RDF can also be achieved through a mapping (e.g. R2RML [Da12]) but these types of mappings are not focused upon in this thesis.

A problem common to all mapping types is that their creation can be a time consuming task [Fa11] often requiring the input of knowledge from domain experts [Ra11]. Therefore, mapping research has recognised that it would be advantageous to be able to reuse previously created mappings for the purpose of reducing the time it can take to create new mappings [Gr11]. **Mapping reuse** consists of discovering and sharing specific existing mappings [Th13], discovering information from existing mappings to help with the creation of new mappings [Zh06] and discovering and using the properties from existing mappings to aid in the creation of new mappings [Au05, Vo09].

Datasets can evolve and change [Um10] as new versions are periodically released or updated over time. This evolution can cause changes to the representation of vocabulary terms and instances within a dataset. When a change occurs to a vocabulary term or instance, its identifier (URI) may also change causing the old identifier to no longer dereference [Po11]. This can cause problems as, per the Linked Data principles [Bi09], URIs should be used to identify data and the URIs should be dereferenceable and provide information about the data. With regard to mappings, a change to the identifier of the representation of a vocabulary term or instance can cause mappings, which reference the old identifier, to become invalid [Do15a]. An invalid mapping (as defined in this thesis) is a mapping that no longer produces correct results due to changes in the representation of vocabulary terms or instances in the datasets referenced by the mapping. Invalid mappings produce incorrect results, cause interoperability issues and hence reduce the quality of the dataset where they are employed. Therefore it is important to discover invalid mappings so decisions can then be made to repair the invalid mapping or to retire it. The discovery and repair of invalid mappings are part of **mapping maintenance** [Do15]. Mapping maintenance is a time consuming task [Do15], similar to the creation of new mappings, and it is compounded further when considering

---

\(^4\) This category of mapping is used to transform instance data that is represented according to the vocabulary of one dataset into the vocabulary of another dataset (see Section 2.2.2).

\(^5\) This category of mapping is used to link related instances across Linked Data datasets (see Section 2.2.3).
the number of mappings that exist between some Linked Data datasets [Ne16]. For example, the DBpedia dataset has over 27 million mappings (interlink category) to external datasets [Le15].

Existing approaches that support mapping maintenance or mapping reuse vary in the type of features they support. For example, DyKOSMap [Do15a] is a tool specifically designed to support mapping maintenance that does not support the reuse of mappings. COMA++ [Au05] is an ontology matching tool, which supports aspects of mapping reuse and maintenance, but does not support the discovery of invalid mappings. MooM [Th11] is a mapping retrieval tool that only supports the discovery of specific mappings. To the author’s knowledge, there exists no single methodology or tool that supports both mapping maintenance and mapping reuse. In addition (to the authors knowledge), no set of common practices or principles exist to support users who wish to perform both. Since mapping maintenance and mapping reuse are not reliant upon each other, one can be performed without performing the other. However, mapping maintenance does complement mapping reuse. Since mapping reuse is concerned with reusing existing mappings, it is important to ensure those mappings to be reused are not invalid (mapping maintenance is concerned with discovering invalid mapping).

In this thesis, a methodology named SPARQL Usage for Mapping Maintenance and Reuse (SUMMR), is proposed and evaluated. The aim of SUMMR is to provide a methodology to support users in performing maintenance and reuse of Linked Data mappings. SUMMR is a query-based methodology for mapping maintenance and reuse. A query-based methodology for mapping maintenance and reuse relies on a query language to express mapping maintenance and reuse tasks and a query processor to execute those tasks over mappings. It is envisaged that a query-based methodology would be beneficial as users would be more familiar with a query language to express tasks as opposed to an ad-hoc language. In addition, users would likely have the software deployed (query processor) to execute the tasks - reducing the need to integrate ad-hoc software. Existing approaches for data retrieval and consumption in the Linked Data field [Fa14, Ra12, Sc14a] provide support for users to discover and retrieve information from different datasets through the use of query templates. The query templates provide support for non-technical users and/or users unfamiliar with the structure of a database, in order to discover and retrieve this information. Given this demonstrated utility of query templates, SUMMR provides query templates, which rely
on standard SPARQL queries, to support users in performing mapping maintenance and reuse tasks.

1.2. Research Question

The research question investigated in this thesis is:

*To what extent can a query-based methodology improve the range of Linked Data mapping maintenance and reuse tasks compared to the state of the art?*

1.2.1. Definitions

The definition of the terms used in the research question and definitions used in the wider context of this thesis are explained here.

A *methodology* is defined as a body of methods, rules and postulates employed by a discipline [Ie00].

A *mapping* as defined by Euzenat and Shvaiko [Eu07] is “…the oriented, or directed, version of an *alignment*: it maps entities [or resources] of one ontology [or dataset] to at most one entity [or resource] of another ontology [or dataset]”. An *alignment* [Eu07] consists of resources from source and target datasets, along with their *correspondences* - which is a relation between the source and target resources (such as equivalent, not equivalent, etc.). An *alignment* describes the relationship between source and target resources and a *mapping* is what enforces (or executes) that relationship (such as a set of rules or a query). The terms *entity* and *resource* are used interchangeably in this thesis.

The definition of *mapping maintenance* defined by Dos Reis et al. [Do15] is adopted in this thesis, which is: “[mapping maintenance is]…the task aiming to keep existing mappings in an updated and valid state, reflecting changes affecting KOS’s [Knowledge Organisation System] entities at evolution time”. Dos Reis et al. further refine this definition of mapping maintenance into four sub-problems:

- The identification of changes in KOS resources defining mappings that can potentially invalidate them.
- The decision concerning the impact of the identified KOS changes on the validity of mappings.
- The determination of the appropriate actions to apply to the mappings to restore their validity.
• The implementation of specified actions for executing modifications on mappings.

It is important to note that in the literature, the definition of a mapping differs. For example the definition of a mapping by Dos Reis et al. [Do15] is not the same as the definition of that by Euzenat and Shvaiko [Eu07]. The Dos Reis et al. definition of a mapping is equivalent to the definition of a \textit{correspondence} by Euzenat and Shvaiko (described above). Given this difference between the two definitions of a mapping, the author of this thesis believes that the definition of mapping maintenance by Dos Reis et al. is still appropriate, as changes to correspondences can have an effect on alignments which in turn can have an effect on mappings.

To the knowledge of the author of this thesis, a formal definition for \textit{mapping reuse} does not exist. The definition of mapping reuse used in this thesis was created by the author and was derived from the examination of existing approaches that support the reuse of mappings in some manner. \textit{Mapping reuse} is defined as the use of existing mappings in three situations:

• Specific mappings can be discovered and shared for direct use [Th13]. Sharing mappings involves sending mappings to or publishing mappings for others to use.

• Information from existing mappings can be discovered to provide insight to help with the creation of new mappings [Zh06]. Such information could be discovering the exact terms from a dataset’s vocabulary that must be used and whether any values need to undergo a transformation (converting minutes to seconds or miles to kilometres) in order to be valid according to a dataset’s vocabulary. Another example would be discovering other vocabulary terms that are used in a mapping. If a mapping is searched for based on the term ‘film’ - what other vocabulary terms associated with ‘film’ also appear in that mapping - duration, producer, director, actors and so on.

• The properties from existing mappings can be directly re-used in the creation of new mappings [Au05, Vo09]. The \textit{properties of a mapping} refers to the details described within a mapping such as the source and target resources involved and any data transformations associated with a resource (for example: converting miles to kilometres or concatenation of a first name and a last name). Meta-data can also be associated with a mapping [Th13] and associated meta-data is also treated as a mapping property in this thesis.
Mapping maintenance and reuse *tasks* are defined as five individual tasks that should be supported by an approach for both mapping maintenance and reuse. The five tasks are derived from mapping maintenance and reuse use cases which were identified from the state of the art review in this thesis. See *Chapter 3* for a description of these use cases and how the tasks were derived.

The terms *ontologies*, *vocabularies*, and *datasets* occur in this thesis. An *ontology* is a set of assertions that are meant to model a particular domain of knowledge [Eu07]. Data in ontologies can be broken into two components: the Tbox (terminological component) and Abox (assertion component) [Gu16]. The Tbox contains the declaration of classes and properties which are used to describe knowledge within a domain, along with rules, constraints and reasoning axioms about those classes and properties. The Abox consists of instance data (described according to the classes and properties from the Tbox). In this thesis, the term *vocabulary* refers to a Tbox component that consists of the declaration of classes and properties but does not consist of rules, constraints or reasoning axioms. A vocabulary could be considered as a lightweight Tbox component. The term *dataset* is used to describe a collection of information. Specifically for RDF (and hence Linked Data) datasets this refers to a collection of RDF triples. A dataset is a meaningful collection of triples that deals with a certain topic, originate from a certain source or process, are hosted on a certain server, or are aggregated by a certain custodian [Al09]. A dataset, can consist of both Tbox and Abox components, or just Abox components.

1.2.2. Research Objectives

In order to address the research question outlined above, the following research objectives were identified for this thesis:

- **RO1**: Perform a state of the art review of existing approaches for mapping maintenance or mapping reuse, or both, and existing approaches for representing ontology mappings.
- **RO2**: Design a query-based methodology for undertaking both mapping maintenance and reuse. This will include an appropriate mapping representation\(^6\) and *query templates* to perform maintenance and reuse.
- **RO3**: Design a prototype software tool that implements the new query-based methodology from RO2.

\(^6\) A mapping representation defines how a mapping is modelled, described and encoded.
• **RO4:** Evaluate the query-based methodology from RO2 in lab-based experiments. This will involve evaluating (i) the expressivity of the mapping representation used by the methodology, and (ii) the ability of the query templates to support mapping maintenance and reuse tasks.

• **RO5:** Evaluate the utility of the query-based methodology from RO2 in a case study, where the software tool from RO3 will be deployed in the release process of a real world Linked Data dataset.

### 1.2.3. Thesis Contributions

The major contribution of this thesis is the design and development of the **SPARQL Usage for Mapping Maintenance and Reuse (SUMMR)** methodology, for performing mapping maintenance and reuse of two categories of Linked Data ontology mappings (mappings to transform data between vocabularies of Linked Data datasets and interlinks between instances of Linked Data datasets). SUMMR provides a set of practices, through the **elements of its design** and a set of procedures and rules in the form of **query templates** for performing mapping maintenance and reuse. No other methodologies exist explicitly for performing both mapping maintenance and reuse of Linked Data mappings. Evidence from experiments indicate that SUMMR templates can perform all mapping maintenance and reuse tasks (derived from the state of the art in **Chapter 3**) over mappings represented in SPARQL Centric Mapping Representation (see **Section 5.2.1**). The case study conducted shows the usefulness of SUMMR in a real world situation where it has been used to check for invalid interlink category mappings in the live management process of the DBpedia dataset. The use of SUMMR had a direct impact on the quality of the DBpedia v.2015-10 release through the detection of invalid interlink category mappings so they could be repaired or removed from the dataset. SUMMR advances the state of the art in mapping maintenance and reuse through the creation of a new taxonomy, consisting of five mapping maintenance and reuse tasks and six use cases. The use cases were identified from a review of existing approaches that support mapping maintenance and mapping reuse. Two of the use cases are mapping maintenance use cases and the remaining four are mapping reuse use cases. The tasks were derived from these use cases and they represent the individual tasks that need to be supported to perform the use case. There are approaches from the state of the art with support both mapping maintenance use cases such as DyKOSMap [Do15a] and LogMap [Ji11], however neither of these two approaches also support any of the reuse
use cases. In terms of mapping reuse use cases, two existing approaches - LinkLion [Ne14] and BioPortal [No08] support three out of four of these use cases but support no maintenance use cases. Only one approach from the state of the art supports use cases from the maintenance and reuse category. This approach is COMA++ [Au05] which supports one maintenance use case and two reuse use cases. No single existing approach from the state of the art support performing all six of these use cases and hence, and all five of the tasks. The SUMMR methodology advances the state of the art as it does support performing all use cases and tasks. As well as advancing the state of the art, it is envisaged that the SUMMR methodology will also be of benefit to the providers and maintainers of Linked Data datasets who wish to discover and repair invalid mappings. It is also envisaged that SUMMR will also benefit dataset providers and maintainers who wish to discover and create new mappings related to datasets of interest, through the reuse of existing mappings.

A minor contribution of this thesis is the mapping representation that was developed as part of the SUMMR methodology, named the SPARQL Centric Mapping Representation (SCMR). The SCMR is an RDF-based mapping representation for representing that is designed to represent two categories of Linked Data ontology mappings (mappings that transform data between vocabularies and mappings that link semantically similar instances between Linked Data datasets) at a fine grain level. The granularity of a mapping refers to how properties of a mapping are modelled and described. A fine grain mapping refers to a mapping where all the properties of a mapping are modelled and described explicitly in a knowledge representation format. A coarse grain mapping is the opposite where the entirety of the mapping is modelled as a single string. SCMR, whether used with or without SUMMR provides a way to represent ontology mappings and also allows for mapping meta-data annotation. SCMR provides support for mapping maintenance and reuse through facilitating fine grained discovery and retrieval of a mapping and its properties. It is envisaged that its fine grained nature will be useful in other applications (beyond mapping maintenance and reuse), such as mapping governance [De15], that require retrieval or analysis of mappings.

The second minor contribution of this thesis is an open-source software tool named the SUMMR Interlink Validation Tool7. The SUMMR Interlink Validation Tool is an open-source, java-based command line tool that implements SUMMR for the discovery

7 https://github.com/aligned-h2020/ALIGNED_Code/tree/master/interlink_validation_tool
of invalid interlink category mappings. The tool was used in the case study of this thesis to validate the interlink category mappings that were published in the v.2015-10 DBpedia dataset release (no invalid interlink detection has been conducted before in the dataset). The tool is now part of multiple technologies used by the DBpedia dataset administrators and has been incorporated into the H2020 ALIGNED project’s DBpedia Phase 2 Trial Platform. The SUMMR Interlink Validation Tool would be useful to other Linked Data datasets administrators, that need to maintain interlink category mappings from a source dataset to multiple target datasets. The administrators can use the tool to discover if any of the interlink category mappings they maintain, have become invalid as the source dataset or any target datasets evolve.

There have been three publications associated with the research in this thesis to date:


  This publication presents the SUMMR methodology in detail and presents the evaluation related to the mapping maintenance aspect of the SUMMR methodology which is presented in Section 6.4 in this thesis. This publication also presents the SUMMR methodology case study, which applied SUMMR to the DBpedia interlink management process. That case study is presented in Section 6.5 in this thesis.


  This publication presents a preliminary version of the SUMMR methodology, focusing on the mapping reuse aspect of the SPARQL-based methodology. This publication also presents an initial evaluation related to the mapping reuse aspect of the SUMMR methodology which is presented in Section 6.3 in this thesis.


---

8 See http://www.aligned-project.eu

This publication presents an initial version of the SPARQL Centric Mapping Representation (SCMR), developed as part of the SUMMR methodology. The publication also presents the evaluation related to the SCMR which is presented in Section 6.2 in this thesis.

In summary, the major contribution of this thesis is the SUMMR methodology. The minor contributions are the SPARQL Centric Mapping Representation (SCMR) and the open-source SUMMR Interlink Validation Tool.

1.2.4. Assumptions and Constraints

The assumptions and constraints of the research in this thesis are outlined below:

- This research is concerned with performing mapping maintenance and reuse over two categories of Linked Data ontology mappings. These two categories are transforming data described from one vocabulary into another (see Section 2.2.2) and for interlinks between similar instances in different datasets (see Section 2.2.3). Two other categories of mappings which are not considered in this research are those which are used to transform (uplift) data from a non-RDF form into RDF and transform (lower) data from RDF to another form (see Section 2.2.1).

- The research in this thesis can be applied to ontology mappings (from the two categories described in the previous bullet point) that would be derived from both simple and complex correspondences (see Section 2.2.1 for definition of simple and complex correspondences).

- This research is not concerned with the automatic repair of mappings as this is beyond scope. Rather, the research is concerned with how existing mappings can be altered in order to repair it.

- This research is not directly concerned with the matching phase of ontology mapping creation and therefore not concerned with the decision making around establishing the semantic relationship for resources from ontologies. This is a large area of research in its own right and can involve complex processes such as discovering these semantic relationships from term resource class and property
hierarchies and enumerated types. Ontology matching tools are commonly employed in this task to aid human experts. However, while not directly concerned, the research in this thesis could be used in conjunction with ontology matching tools to supplement them with mapping maintenance and reuse support.

- This research is concerned with the use of standard SPARQL queries (queries that adhere to the SPARQL standard specification set out by W3C) to perform mapping maintenance and reuse tasks and use cases over mappings. This provides the advantage of performing the mapping maintenance and reuse tasks and uses on any standard SPARQL processor.

- Since SPARQL is designed to operate over data encoded in RDF, this research is particularly interested in mappings that are encoded in RDF.

1.3. Research Methodology

Initially, a state of the art review of existing approaches that support mapping maintenance, mapping reuse or both was undertaken. Six mapping maintenance and reuse use cases were identified. From an analysis of these six use cases, five mapping maintenance and reuse tasks were derived. These five tasks, derived from the six use cases, are what needs to be supported by an approach for mapping maintenance and reuse, in order to perform the use cases. The mapping maintenance and reuse use cases were refined further (see Chapter 4) and detail how a query-based methodology would perform those use cases. The SUMMR methodology was designed, based on the state of the art review and from the six refined use cases in Chapter 4.

The SUMMR methodology includes the specification of a mapping representation - called the SCMR and provides query templates for performing mapping maintenance and reuse. SCMR was designed to be a fine grained RDF-based mapping representation. The SCMR complements the query templates provided by SUMMR through allowing fine grain discovery and retrieval of a mapping and its properties. The SUMMR methodology provides two sets of query templates that use standard SPARQL queries. One set of queries are for performing mapping maintenance and reuse tasks and the other set for performing mapping maintenance and reuse use cases. While SUMMR provides the SCMR to represent mappings, it is not mandatory to use. Multiple RDF-based mapping representations can be used with SUMMR. In this manner, SUMMR templates are specialised for different mapping representations. A software tool that
implements SUMMR was developed named the SUMMR Interlink Validation Tool. Through consultation with the DBpedia dataset maintainers at the University of Leipzig in Germany, requirements for the tool were gathered. The tool facilitates SUMMR’s invalid interlink category mapping detection. It is an open-source, java based command line tool that is used to discover invalid interlink category mappings between a source Linked Data dataset and multiple target Linked Data datasets.

Evaluation of SUMMR was undertaken through the execution of three lab-based experiments and a case study. The expressivity of the SCMR was evaluated in the first experiment described in this thesis (Section 6.2). SUMMR query templates were evaluated with respect to the support they provide to a user to perform mapping maintenance and reuse tasks, described in the second (Section 6.3) and third (Section 6.4) experiments in this thesis. The usefulness of the SUMMR methodology in a real world Linked Data dataset management situation was evaluated through the case study (Section 6.5) described in this thesis.

The next section provides more detail of the evaluation strategy used.

1.4. Evaluation Strategy

The strategy used to evaluate the SUMMR methodology was as follows:

- **Identify individual mapping maintenance and reuse tasks:** From reviewing existing approaches that support mapping maintenance and reuse, five individual tasks that these approaches perform were derived. These tasks were then used to ensure SUMMR query templates could perform them in two evaluations of the SUMMR methodology.

- **Evaluate the expressivity of the SCMR:** This evaluation tested the capability of the SCMR to represent ontology mappings that transform data between vocabularies. The SCMR was compared to the R2R Mapping Language [Bi10] in the evaluation. The creators of the R2R Mapping Language created and published a set of 72 mappings (vocabulary transformation category) between Linked Data datasets. These mappings were represented in SCMR and executed, over sample data that was collected from data sets the mappings reference. The mappings in their original representation (R2R Mapping Language) were then executed over the sample data. The execution results of the mappings represented in SCMR and in the R2R Mapping Language were then directly compared. It was found that the execution results from the two mappings were
identical. This evaluation provides evidence that the SCMR and R2R Mapping Language are equally as expressive when it comes to representing mappings. What separates the two is that the R2R Mapping Language is not specified to represent interlink category mappings, whereas the SCMR is designed to represent another category of mapping too - mappings that link instances across datasets.

- **Evaluate SUMMR’s ability to support a user to perform mapping reuse tasks:** This evaluation tested the effectiveness of the SUMMR query templates support for performing mapping reuse tasks. This was achieved through using SUMMR templates to discover mappings and mapping properties based on mapping retrieval use-cases. Two specialisations of SUMMR templates were used, one specialisation for execution over mappings represented in SCMR and the other for mappings represented in the R2R Mapping Language. The two sets of mappings used in experiment 1 are used in the evaluation (72 mappings represented in SCMR and the R2R Mapping Language). The common information retrieval metrics of precision and recall were used in this experiment. They were used to evaluate the ability of the SUMMR templates to perform the mapping retrieval use-cases. It was found that the SUMMR query templates specialised for SCMR could correctly perform all the mapping retrieval use-cases for the mappings represented in SCMR. However the SUMMR templates specialised for the R2R Mapping Language could not perform all the mapping retrieval use-cases. This was found to be caused by the coarse granularity of how the R2R Mapping Language represents mapping properties. This evaluation provides evidence that SUMMR will support performing mapping reuse tasks (as retrieval is at the core of any reuse task) but the effectiveness of a SUMMR template is effected by the granularity of the mapping representation that it is performing over.

- **Evaluate SUMMR’s ability to support a user to perform mapping maintenance tasks:** This evaluation tested the effectiveness of SUMMR templates to support performing mapping maintenance tasks. Two sets of mappings were used in this evaluation: (i) the set of 72 mappings (of the vocabulary transformation category) represented in SCMR and (ii) a set of mappings consisting of 43 mappings (of the interlink category) between the DBpedia dataset and the DailyMed dataset represented in SCMR. Two gold
standard datasets were created (see Section 6.4.3 for description of datasets and how they were created). The first gold standard dataset contained invalid mappings (from both the vocabulary transformation and interlink category) from the two datasets used in the experiment. The second gold standard dataset was created after the first and it contained the execution results of the repaired versions of the invalid mappings found from the first gold standard dataset. Precision and recall were used as metrics to compare the results of applying the SUMMR template to discover invalid mappings, to the results in the first gold standard dataset. The mappings that were repaired using the appropriate SUMMR template to repair invalid mappings, were executed and their execution results were directly compared to the second gold standard dataset. It was found that the SUMMR query templates could be used to correctly discover all invalid mappings and could successfully repair them when a new vocabulary term or dataset instance is provided to replace the invalid ones in a mapping. This evaluation provides evidence that SUMMR templates support performing mapping maintenance tasks.

- **Evaluate the usefulness of the SUMMR methodology in a real world Linked Data dataset management situation:** This evaluation was undertaken as a case study where SUMMR was applied to the live DBpedia dataset release process to validate interlink category mappings for a new release of the dataset. In the DBpedia dataset release process, interlink category mappings to be used in a new release are not checked to ensure which are valid and which are invalid. This means that invalid interlink category mappings could be published in a new release. The SUMMR Interlink Validation Tool was used to validate 1,679,634 interlink category mappings that were to be published in the v.2015-10 DBpedia dataset release. All source instances within the interlink category mappings were checked against the DBpedia dataset and target instances were checked against their external dataset when possible to access that external dataset through a SPARQL federated call. An interlink category mapping was classed as invalid if the source instance of the mapping did not exist in the v.2015-10 DBpedia release or a target instance of the mapping did not exist in the external datasets (where mappings were checked externally). The SUMMR methodology, via the SUMMR Interlink Validation Tool had a direct impact on the quality of the
DBpedia v.2015-10 release by discovering that 53,418 of these interlink category mappings were invalid.

In summary, the first three evaluations were lab-based experiments to test the SUMMR methodology features, i.e. SCMR and the query templates. The final evaluation used a case study approach to evaluate the usefulness of the SUMMR methodology in a real world Linked Data management situation at a large scale.

1.5. Thesis Overview

The remainder of this thesis is structured as follows:

Chapter 2: Background

This chapter presents useful context for readers of this thesis. It begins with information about the processes typically involved in the creation of mappings. It then describes four different categories of mappings used within the Linked Data domain: mappings to transform data from a non-RDF form to RDF and vice versa; mappings to transform data between different Linked Data vocabularies and mappings that link similar instances across Linked Data datasets.

Chapter 3: State of the Art

This chapter provides a review of existing approaches for mapping maintenance and reuse and a review of existing mapping representations. It identifies six mapping maintenance and reuse use cases and five tasks. The six use cases are further refined in Chapter 4, which influenced the design of SUMMR and the five tasks are used in the evaluation of the SUMMR templates in Chapter 6.

Chapter 4: Refinement of Use Cases for a Query-Based Methodology for Mapping Maintenance and Reuse

This chapter refines the set of six mapping maintenance and reuse use cases identified in Chapter 3. These refined use cases detail how a query-based methodology would perform the mapping maintenance and reuse use cases.

Chapter 5: Design of the SUMMR Methodology

This chapter describes the design of the SPARQL Usage for Mapping Maintenance and Reuse (SUMMR) methodology. An overview of SUMMR is initially presented. Then the elements of the SUMMR design are described. This details the SCMR and the
SUMMR query templates (specialised for SCMR) and examples are provided. This chapter also discusses how a user would apply the SUMMR methodology to perform mapping maintenance and reuse. Reflections upon the design of SUMMR are also presented, discussing limitations of the methodology and thoughts on the generalisation of SUMMR for application of the Linked Data domain.

Chapter 6: Evaluation

This chapter describes three experiments and a case study as outlines in Section 1.4.

Chapter 7: Conclusion

Here, key findings are presented from the research described in this thesis. It discusses to what extent the research question of this thesis has been answered and the extent to which the research objectives have been met. Possible directions for future work related to the research in this thesis are also outlined.
2. Background

This chapter presents background information related to the research in this thesis that will aid a reader that is unfamiliar with the data mapping domain. Presented in Section 2.1 are the processes of ontology matching and mapping, which are the processes concerned with increasing interoperability between heterogeneous datasets. Section 2.2 presents four different types of mappings in the area of Linked Data.

2.1. Ontology Matching and Mapping

This section briefly describes the processes of ontology matching and ontology mapping [Eu07]. These two processes are carried out between heterogeneous ontologies or datasets. The purpose of these processes is to discover semantically similar resources between the heterogeneous datasets (matching) and establish a means to increase interoperability between them (mappings). The research in this thesis is concerned with the maintenance and reuse of existing mappings. Therefore this section will provide more context as to how those mapping are typically created.

2.1.1. The Matching Process

The ontology matching process, depicted in Figure 2-1 below, takes two inputs and produces one output. The two inputs can be in the form of source and target ontologies [Sh05]. The matching process aims to discover correspondences, which is a semantic relationship (such as equivalent, not equivalent, more general than, less general that etc.), between the entities or resources of the source and target inputs. The output of the matching process is a set of alignments, which consist of entities or resources from the source and target inputs and the correspondence which holds between them. Correspondences fall into two categories which are simple and complex [Gi13]. A simple correspondence is one that relates one entity or resource from an input to one entity or resource of the other input (1:1 relationship). A complex correspondence is one that relates one or more entities or resources from an input to one or more entities or resources of the other input (1:n, m:1, mn relationships).
There are many different software tools in existence which are designed to support the ontology matching process [Sh13, Ot14]. These employ different techniques for discovering correspondences between resources in ontologies. For example, two such techniques are lexical and structural comparison techniques. Lexical comparison techniques involve comparing the names, labels and descriptions of resources from ontologies for semantic equivalence, and often employ auxiliary information such as synonym tables, domain specific taxonomies or general taxonomies such as WordNet [Mi95]. Structural comparison techniques involve representing an ontology in a hierarchical structure, usually in the form of a graph, to easily group classes and sub-classes, and properties together. This makes it easier to detect structural similarities between two ontologies, with the intuition being that if the structure of two resources from different ontologies are similar, then this increases the likelihood that these resources are overall similar. Some tools provide a graphical user interface to visually display ontologies along with established correspondences [Fa11, Au05]. Such applications typically provide a drag and drop environment where resources from ontologies can be linked together and correspondences established between them.

2.1.2. The Mapping Process

The ontology mapping process is concerned with converting alignments (created during the matching process) into mappings, where a mapping is defined as [Eu07]: “…the oriented, or directed, version of an alignment: it maps entities [or resources] of one ontology [or dataset] to at most one entity [or resource] of another ontology [or dataset]”. Depending on the task at hand, different mappings will be used to best suit that task. For example, mappings can be used in a Linked Data integration scenario [Sc12], where data described by the vocabularies of multiple different datasets is to be transformed to a single (canonical) vocabulary.
While not all ontology matching tools contain the functionality to support the creation of mappings from alignments, some such tools do exist [Sc12, To17, Vo09].

2.2. Linked Data Mappings

In this section, four different categories of mappings are described - uplift and lowering mappings, vocabulary mappings and interlink mapping. The research in this thesis focuses on the maintenance and reuse of vocabulary and interlink categories. However, uplift and lowering mappings are also described to indicate the purpose of other types of mappings.

2.2.1. Uplift and Lowering Category of Mappings

Uplift [Bi12, Da12, Di14, Co15] mappings are concerned with transforming data encoded in a non-RDF form (e.g. relational data, XML, CSV, TSV) into an RDF form. The mappings specify how the non-RDF encoded data is retrieved i.e. which table and column from relational data to access or the path through which XML data is accessed. This non-RDF encoded data is the source of the mapping. The target of the mapping is concerned with specifying how the source data is encoded in RDF, how to structure it and the vocabulary terms to use to describe the data.

Lowering mappings [Bi12] do the opposite of uplift mappings. It transforms data encoded in RDF into a non-RDF encoded form. For lowering mappings, the source of the mapping specifies how the RDF data is accessed and the target of the mapping specifies how that source data should be encoded in the chosen non-RDF form. Figure 2-2 depicts an example of XML data that is lifted into RDF and lowered back to XML.

![Figure 2-2. Example of data uplifted from XML to RDF and lowered back to XML [Bi12].](image)
2.2.2. Vocabulary Category of Mappings

The vocabulary category of mappings [Bi10, Ko13] are concerned with transforming instance data from one dataset, and described according to the vocabulary of that dataset, to describing it according to the vocabulary of another dataset. These mappings are used in Linked Data integration situations [Sc12]. They are used to transform data from multiple datasets, described by multiple heterogeneous vocabularies, into a single vocabulary. The intuition behind this is that by having the data in a more homogeneous form, it will be easier to access and manage. Figure 2-3 presents an example what a vocabulary is concerned with. In the figure, data about the Goodfellas movie, described according to the DBpedia vocabulary is transformed to be described according to the Linked Movie Database vocabulary.

2.2.3. Interlink Category of Mappings

Interlinks [Vo09] are used to link semantically similar instances across Linked Data datasets and are part of principle 4, of Tim Berners-Lee’s four Linked Data principles [Bi09]. The term interlink is used in this thesis to specify links that go from (instances) of one dataset to (instances) of a separate external dataset.

An interlink is a single RDF triple where: the subject of the triple interlink is an instance from a source dataset, the object of the triple is an instance from a target dataset and the predicate of the triple is the relation between the two instances. The purpose of establishing interlinks is so instances within specific datasets can be supplemented with additional data (about the same instance) from external datasets. Figure 2-4 displays an example of four interlinks from the Goodfellas movie in the DBpedia dataset, to the same movie in the Linked Movie Database, FreeBase, Opencyc [Ma06] and YAGO [Su07b] datasets.
Four Interlink category of mappings for the movie Goodfellas from the DBpedia dataset to Goodfellas in the Linked Movie Database, FreeBase, OpenCyc and YAGO datasets.

![Image](prefix:owl:<http://www.w3.org/2002/07/owl#>


![Image](http://dbpedia.org/resource/Goodfellas> owl:sameAs <http://rdf.freebase.com/ns/m.0h6r5>. 


Figure 2-4. Examples of four interlinks from the DBpedia dataset to the Linked Movie Database, FreeBase, OpenCyc and YAGO datasets.

2.3. Chapter Summary

The aim of this chapter was to make a reader, who may be unfamiliar with data mappings, gain an understanding of the processes typically involved in their creation and provide examples of different categories of mappings typically uses with the Linked Data domain.

The process of ontology matching was briefly described, which is concerned with discovering correspondences between ontology resources and ultimately producing a set of alignments. The ontology mapping process was also briefly discussed, which is concerned with creating mappings from alignment produced from the matching process. Four different categories of mappings were examined and an example of each type of was presented.
3. State of the Art

This chapter presents the state of the art of approaches for mapping maintenance, mapping reuse and mapping representation. Section 3.1 presents an introduction to the chapter and provides formal definitions of mapping maintenance and mapping reuse. Section 3.2 reviews thirteen existing approaches for mapping maintenance and reuse. Section 3.3 reviews eight existing approaches for representing Linked Data ontology mappings. Section 3.4 consolidates the mapping maintenance and reuse use cases that were identified by the author during the review of the existing approaches in Section 3.2 and then derives the individual tasks needed to perform the use cases. Section 3.5 presents an analysis of the extent to which existing mapping maintenance and reuse approaches support the use cases and tasks identified. Also presented is an overall comparison of the existing mapping representations with respect to selected characteristics. Section 3.6 finishes the chapter with a brief summary.

3.1. Introduction

The purpose of this chapter is to provide a state of the art review of the research in the fields of mapping maintenance, mapping reuse and mapping representation. For this review, existing approaches that support mapping maintenance, mapping reuse or both are examined. These approaches are examined for the mapping maintenance or reuse use cases they support. Existing approaches for mapping representation are reviewed on their characteristics such as: what category of mapping they represent (vocabulary transformation and/or interlinks), whether they support annotation with meta-data, how they are encoded and if they are executable.

3.2. Existing Approaches for Mapping Maintenance and Reuse

This section lists fifteen existing approaches from the state of the art literature that support either mapping maintenance, mapping reuse or both. In the view of the author, this is an exhaustive list of existing approaches that are relevant to the research in this thesis. The approaches are analysed to determine what mapping maintenance and reuse use cases they support and these use cases are summarised as bullet points at the end of each sub-section.

22
3.2.1. DyKOSMap

The DyKOSMap (Dynamic Knowledge Organisation System Mapping) [Do15a] framework is for the semi-automatic detection and repair of invalid alignments between datasets. DyKOSMap does this in a three step approach which first involves (i) dataset evolution, followed by (ii) mapping interpretation and finishes with (iii) mapping adaptation.

The dataset evolution step involves performing an analysis between two different versions of a dataset to detect changes between the two caused by dataset evolution. DyKOSMap employs the COnto-Diff (Complex Ontology Diff) [Ha13] algorithm for dataset change detection. COnto-Diff operates by performing an ontology matching operation, using the GOMMA tool [Ki11], over two versions of a dataset which results in a set of alignments between the two versions. Taking the two dataset versions and their alignments as input, a series of rule-based algorithms are used to detect both syntactic and semantic changes between the two versions of the dataset. The output of the COnto-Diff algorithm are “diff evolution mappings” which contain the changes between the two versions of the dataset.

The mapping interpretation step involves interpreting (analysing) the alignments (that are checked to determine if they are valid or invalid) to discover and retrieve the source and target dataset resources within the alignment.

The mapping adaptation step involves detecting which alignments are affected by dataset evolution and the repair of those alignments where possible. First, affected alignments are determined by checking if a source or target resource exist in the diff evolution mappings from the dataset evolution step. If it does exist, that means that an alignments source or target resource has changed in some way and the alignment is in need of repair. DyKOSMap defines six mapping adaption actions that it can apply to an alignment for attempted repair. These six actions are: add correspondence, remove correspondence, move correspondence, derive correspondence, modify semantic relation of correspondence and no action. Before applying one of these actions to attempt to repair an alignment, a decision is first made to select the correct actions. This decision is made by a designed decision heuristic which takes into account the affected alignments (from the mapping interpretation step), the diff evolution mappings (from the dataset evolution step) and ontology change patterns [Ha13]. Ontology change patterns describe the types of changes that can occur during dataset (or ontology) evolution. When the adaption actions have been chosen by the decision heuristic, they
are then applied, which alter the alignment for the purpose of attempted repair. The potentially repaired alignments are output to be validated by a user.

DyKOSMap is an advanced framework able to detect invalid alignments caused by both syntactic and semantic changes to resources in a dataset the alignment references. DyKOSMap also provides semi-automatic repair of alignments. There is no mention of a formal alignment representation in the literature on DyKOSMap. DyKOSMap supports two mapping maintenance use cases:

- Discover invalid alignment due to changes in datasets an alignment references.
- Repair of an invalid alignment.

3.2.2. DSNotify

DSNotify (DataSet Notify) [Po11] is a framework designed to detect change events (create, remove, move, update) in Linked Data datasets with the aim to support the maintenance of interlink category mappings between the datasets.

DSNotify employs a monitor component, which periodically accesses a dataset (via SPARQL queries) for dataset instances. The monitor can be configured to search for specific instances (for example, only find instances that are cars, or countries or movies etc.) For each new instance representation it encounters, it creates a feature vector related to that representation.

A feature vector in this case is data associated with an instance representation. In terms of RDF data - this is all the triples associated with an instance representation. The feature vector can be configured to consider the subject, predicate and object of a triple, or just the subject and predicate. Feature vectors are categorised according to (i) recently added data to a dataset, (ii) recently removed data from a dataset and (iii) data removed a longer time ago from a dataset.

Feature vectors from the three categories are monitored to detect change events in a dataset. DSNotify employs a housekeeper component that is used to call an algorithm to detect move and remove change events. To detect move change events, the algorithm compares feature vectors from the categories of (i) recently added data to a dataset and (ii) recently removed data from a dataset. String similarity measures are used to compare the similarity of the feature vectors. If this measure is below a certain threshold value, then the instance representations is classed as having undergone a move and a move change event is issued. To detect remove change events, the algorithm moves all the feature vectors, which surpass a certain threshold time limit, from the (ii) recently
removed data from a dataset category to the (iii) data removed a longer time ago from a dataset category. For each feature vector moved in this way, a remove change event is issued. To detect update change events, the monitor component is used. It compares newly created feature vectors (from its current monitoring activity) to the feature vectors in the (i) recently added data to a dataset category - if available to do so. String similarity measures are used again to compare the similarity of the feature vectors. If the measure is below a threshold value, then the instance representations is classed as having undergone an update and an update move change event is issued. Create events are issued for each new instance representation the monitor component encounters.

DSNotify provides an RDF-based vocabulary – named the DSNotify Eventset Vocabulary for describing and logging change events. The vocabulary describes how why and when instances within datasets change. Figure 3-1 depicts the classes and properties of the DSNotify Eventset Vocabulary.

![Diagram of DSNotify Eventset Vocabulary](image)

Figure 3-1. Classes and properties of the DSNotify Eventset Vocabulary [Po11].

The purpose of DSNotify is to detect change events in Linked Data datasets for the purpose of supporting the maintenance of interlink category mappings between those datasets. DSNotify does not directly detect invalid interlink category mappings, it supports this activity through detection of the events which can cause the mappings to become invalid and logging those events according to the Eventset Vocabulary. A user will use the logs to detect if any mappings (that the user is concerned with) have become invalid. DSNotify partially supports the mapping maintenance use case of:

- Discover invalid interlinks due to changes in datasets an interlink references.
3.2.3. Khattak et al.

The research of Khattak et al. [Kh15] proposes an alignment maintenance approach with the goal to discover invalid alignments and minimise the time required to discover new alignments. This approach consists of two main components: the Change History Log and the Reconciliation of Mappings.

The Change History Log is an RDF-based vocabulary proposed to log all changes that occur to a dataset as it evolves over time. The changes that it is designed to record are the addition, renaming and deletion of classes, properties or instances. In addition, the log also records why these changes were made, when they were made and who or what made them (could refer to a person or a machine). Figure 3-2 displays the core elements of the Change History Log.

![Figure 3-2. Core Elements of the Change History Log [Kh15].](image)

The Mapping Reconciliation component of this approach proposes the use of ontology matching tools to replace invalid alignments and also discover new ones. But instead of running these tools over the entirety of resources in two datasets (which can be time consuming) the tools will only use changed dataset resources that are recorded in the Change History Log. The Mapping Reconciliation component defines a procedure that first discovers invalid alignments by checking if the dataset resources that an alignment references have changed through checking the Change History Log. If a resource an alignment references has been recorded as changed in the Change History Log, then that alignment is classed as invalid and is deleted. According to Khattak et al, only newly added or changed resources need to undergo the matching process and these resources exist within the Change History Log. The next step of the procedure uses a SPARQL query to retrieve newly created or changed resources from the Change History Log.
Log. Then these resources are input to ontology matching tools to discover new alignments.

The approach by Khattak et al. relies on recorded changes to resources from a dataset in the Change History Log. These recorded changes can then be used to discover invalid alignments by checking if alignments reference any of the changed resources. This approach does not repair invalid alignments, instead it deletes invalid alignments and attempts to discover new alignments. The approach by Khattak et al. supports the mapping maintenance use case of:

- Discover invalid alignments due to changes in datasets an alignment references.

3.2.4. Maveric

Maveric (Mapping verification) [Mc05] is an application for the detection of invalid mappings between data integration systems. It operates in two main phases: a training phase and a verification phase.

In the training phase, sensors are set up for a dataset. A number of queries are then established to probe the dataset. The dataset that is being probed has a number of existing mappings established to external datasets that are known to be correct at the time of the probing. Therefore the probing queries are set up to retrieve data that is reliant on data from the external datasets (via the established mappings). The results from the probing queries are used as training data. Maveric expands the training data by artificially introducing errors to the probing query results. This combined training data is then used to train the sensors for a dataset. The sensors are trained so when they are used in the validation phase, they will be able to detect variations in expected query results. This variation could be caused by mappings being no longer valid.

The verification phase is concerned with verifying that the existing mappings are still valid. This is done by periodically querying the dataset (for data what would rely on the established mappings). The results from these queries are then input to the sensors which calculate a score based on the variation of the query results compared to what is expected. If the score is over a threshold value, an alert is sent to a filter module for further processing to determine if the alarm is a correct or false. If an alarm has been deemed (automatically) to be correct, an alert is made to a user about a potentially invalid mapping.

Maveric provides an approach to detecting invalid mappings through learning the expectation of certain query results (which rely upon existing mappings). If there is a
variation in the expected results, an alarm is raised indicating potentially invalid mappings. Maveric only detects invalid mappings, it does not consider the repair of them. Maveric supports the mapping maintenance use case of:

- Discover invalid mappings due to changes in datasets a mapping references.

3.2.5. LogMap

LogMap [Ji11] is an ontology matching tool that also incorporates functionality to detect alignments that have become invalid due to changes in the datasets the alignment references. LogMap does this through modelling (updated versions) of source and target datasets (it models instance data, instance data classes and parent classes in a hierarchy format) and all existing alignments it contains (not just the alignments that reference the source and target dataset under consideration) in a Horn propositional logic representation.

LogMap then runs a Horn satisfiability algorithm over the Horn representation of the dataset resources and the existing alignments. This algorithm checks if the source and target resources referenced in an alignment, are satisfiable with regard to the resources in the source and target datasets. If a referenced resource in an alignment is found to be unsatisfiable, then that alignment is classed as invalid.

LogMap attempts to repair each invalid alignment through a repair algorithm. The repair algorithm will search all existing alignments that contain an unsatisfiable resource, for example the source resource, and examine the target resources of that alignment. It will then check to see if these target resources match any new resources in the updated source dataset. If they do, then this new source resource will replace the old unsatisfiable source resource in the alignment – potentially repairing it (needs to be verified by human user).

LogMap uses a reasoning-based approach for the detection and repair of invalid alignments. LogMap supports two mapping maintenance use cases:

- Discover invalid alignment due to changes in datasets an alignment references.
- Repair of an invalid alignment.

3.2.6. Dependency Model

The research of Boran et al. propose a dependency model [Bo10] approach which supports the detection of invalid alignments that are in use by a data integration system. The dependency model itself is an OWL (Web Ontology Language) ontology. Entire
datasets and alignments and not modelled in the dependency model, but references to the datasets and alignments are modelled. For example, alignments under consideration in this approach are represented in the Alignment format [Eu04, Da11], but the dependency model only models a reference to an alignment. The references of datasets and alignments are modelled as OWL classes. Dependency relationships are then modelled which represent the relationship between specific datasets and specific alignment references. An inference algorithm is then run over the dependency relationships. This inference results in dependency chains. The dependency chains contain all the datasets and alignment references that depend on each other. When a dataset evolves and a particular resource within that dataset changes, the dependency chains can be consulted which will highlight all the existing alignments that are affected by this change. The dependency model approach does not support the detection of changes in datasets itself, it assumes those changes are detected in another process.

The dependency model uses a reasoning based-approach which infers dependency chains between related datasets and alignments. The dependency chains are then consulted, when a dataset changes to find alignments that are potentially invalidated by the dataset change. Therefore this approach supports the mapping maintenance use case of:

- Discover invalid alignments due to changes in datasets an alignment references.

### 3.2.7. WOD-LMP

The Web of Data - Link Maintenance Protocol (WOD-LMP) [Vo09] is a SOAP-based communication protocol designed to be used between source and target Linked Data datasets. Its primary purpose is for the detection of invalid interlink category mappings due to changes in one of the datasets. The protocol supports three features: (i) link transfer, (ii) request dataset change list and (iii) subscribe to dataset changes.

The link transfer feature allows a source dataset to send a set of interlink category mappings to a target dataset, so the WOD-LMP on the target dataset side can keep track of them and also use them to create back links. Considering that an interlink is a RDF triple, where the subject of the triple is a source dataset instance and the object of the triple is a target dataset instance; creating back links requires switching the position of the subject and object of the triple. The interlink category mappings are sent to the target dataset as a Link Notification message. The source dataset can also send a Link
Deletion Notification message to the target dataset to notify the deletion of interlink category mappings.

The dataset change list feature allows a source dataset to request a list of changes that have occurred to instances in the target dataset for a certain time period. The source dataset can then use this list of changes to detect if any existing interlink category mappings were affected by the changes causing them to become invalid. To get the list of changes, the source dataset sends a Get Changes message and the target dataset responds with a Change Notification message.

The subscribe to dataset changes feature allows the source dataset (which has existing interlink category mappings to the target dataset) to send the target instances of a set of interlink category mappings (via a Link Notification message) to the target dataset. The target dataset will monitor these instances and if a change occurs, will notify the source dataset about the changes to specific instances (via a Change Notification message). The source dataset can then use this information to repair an affected interlink category mapping or delete it.

WOD-LMP allows for the detection of invalid interlink category mappings, through a source dataset requesting changes to a target dataset or subscribing to the target data for notifications about changes to particular instances in that dataset. WOD-LMP also support the reuse of existing interlink category mappings through allowing a source dataset to send a set of interlink category mappings to a target dataset and creating back links from those interlink category mappings sent. WOD-LMP supports the mapping maintenance and mapping reuse use cases of:

- Discover invalid interlinks due to changes in datasets an interlink references.
- Discover back links to aid in the creation of new interlinks.

3.2.8. COMA++

COMA++ [Au05] is an ontology matching tool. It contains a repository for storing alignments created by the tool and provides functionality to manipulate those stored alignments.

COMA++ provides functionality to compare two alignments to determine the differences and similarities between them, alignments between common source and target datasets can be merged together and existing alignments can be edited - allowing them to be changed for purposes of repair. It also allows for the discovery of ‘mapping paths’ which can be used in the establishment of new alignments. As an example of a
mapping path, consider the following. If two alignments exist in the repository between the resource ‘car’ from dataset ‘A’ to the resource ‘vehicle’ in dataset ‘B’ and ‘vehicle’ in dataset ‘B’ to the resource ‘four_wheel_vehicle’ in dataset ‘C’. Based on those two alignments, a new alignments between ‘car’ from dataset ‘A’ to ‘four_wheel_vehicle’ in dataset ‘C’ can be inferred. COMA++ also allows alignments to be retrieved and shared. However, since COMA++ represents alignments in a proprietary format, it can only share the alignments with other COMA++ tools.

COMA++ is an ontology matching tool that supports the reuse of alignments through exporting and importing existing alignments to other tools and through the discovery of mapping paths. COMA++ also supports the repair of alignments by allowing existing alignments to be edited. COMA++ supports the mapping maintenance and mapping reuse use cases of:

- Discover alignments for sharing.
- Discover mapping paths to aid in the creation of new alignments.
- Repair of an invalid alignment.

3.2.9. MooM

The Management of Ontology Mappings (MooM) [Th11] framework, is a framework for the reuse of alignments. MooM provides a repository for storing alignments and provides a user interface which allows a user to annotate individual alignments with mapping meta-data terms. The alignments are represented in the Alignment format. The meta-data terms come from the Ontology Mapping, Management and Reuse (OM2R) [Th13] meta-data model. MooM allows for these alignments to be discovered and retrieved. To do this, MooM provides a set of SPARQL queries to allow these alignments to be discovered and retrieved based on specific meta-data terms.

MooM supports the mapping reuse use case:

- Discover alignments for sharing.

3.2.10. LinkLion

LinkLion [Ne14] is an online repository which allows users to submit and store interlink category mappings between Linked Data datasets. The purpose of LinkLion is to facilitate the publication, retrieval and reuse of interlink category mapping. When a user submits interlink category mappings to the repository, they can also specify meta-data - such as the ontology matching tool used in their creation (if an ontology matching tool
was used at all), the matching algorithms used by a matching tool, the source and target datasets of the interlink category mappings and the time and date they were created. Submitted interlink category mappings and accompanying meta-data are represented according to the LinkLion ontology. The LinkLion ontology provides a mapping representation for representing interlink category mappings and their meta-data in RDF and is discussed in further detail in Section 3.3.1. LinkLion allows interlink category mappings (and their meta-data) to be browsed and retrieved through a user interface or through a SPARQL endpoint. Users are free to pose SPARQL queries over the interlink category mappings for analysis, retrieval and reuse purposes. Users could also use LinkLion to discover backlinks and mapping paths but would have to create the SPARQL queries themselves to do so.

LinkLion supports the mapping reuse use case:

- Discover mappings for sharing.
- Discover back links to aid in the creation of new interlinks.
- Discover mapping paths to aid in the creation of new interlinks.

### 3.2.11. BioPortal

BioPortal [No08] is an online repository for biomedical datasets. Users can submit new datasets to the repository, browse existing ones and retrieve entire datasets or instances from datasets. BioPortal also contains an interlink category mapping repository (similar to the LinkLion repository discussed in Section 3.2.10) to store interlink category mappings between these datasets for users to browse and reuse. Users are free to submit interlink category mappings and meta-data to the repository. These interlink category mappings are represented according to the mapping ontology - which is a mapping representation for representing the interlinks and associated meta-data in RDF. This BioPortal mapping is discussed further in Section 3.3.2. The BioPortal interlink category mappings are accessible through SPARQL queries. Users are free to pose queries over them for analysis, retrieval and reuse. Users can also use BioPortal to discover backlinks and mapping paths but similar to LinkLion - would have to create the SPARQL queries to do so.

BioPortal supports the mapping reuse use case:

- Discover mappings for sharing.
- Discover back links to aid in the creation of new interlinks.
- Discover mapping paths to aid in the creation of new interlinks.
3.2.12. **Community Driven Ontology Matching**

The Community Driven Ontology Matching \cite{Zh06} approach proposes a repository for the storage of alignments generated by ontology matching tools. Alignments are represented in the Alignment format, with an extension that allows for meta-data to be provided about the alignment itself and also meta-data for each element of the alignment. For example meta-data can be provided for each source and target resource referenced in an alignment. The purpose of this repository is to contain annotated existing alignments that can be used by ontology matching tools as auxiliary information to help during matching tasks. When additional matching tools make use of the repository to aid in the creation of new alignments, the idea is they submit those new alignments to the repository. Therefore it is being further enriched by the community who make use of it.

The Community Driven Ontology Matching approach proposes a repository with annotated existing alignments which can be used to help with the creation of new alignments. This approach supports the mapping reuse use case:

- Discover information from existing alignments to aid in the creation of new alignments.

3.2.13. **Li et al.**

The research by Li et al. \cite{Li11} proposes an approach aimed at discovering new mappings for meta-querier systems, from reusing existing mappings. A meta-querier system is a data integration system that accesses multiple heterogeneous datasets through accessing individual local query forms. Meta-queriers allow users to enter a single query in a global query form. That query then needs to be reformulated to work in multiple heterogeneous local query forms. This is done through the use of mappings between the global query form and the local query forms.

Li et al. propose the M-Onontology as a way to model the mappings between query forms. The mappings are modelled as a directed acyclic graph. Each mapping instance is made up of three types of nodes, E, G and A nodes, which are model different concepts of the mapping and includes code to reformulate a query. Edges between the nodes denote a relationship between the concepts. E, G and A nodes which are subclass representation of the concepts. For example E nodes may represent the concept of “car” and “jeep”. Those E nodes would then be connected to a G node which would be a more general concept such as “four wheel vehicle”. Then G nodes are connected to A nodes.
which are even more general such as the concept “vehicle”. Li et al. recognise that mappings can become invalid as datasets change over time, and the mappings need to be repaired and evolve to remain valid. The M-Ontology keeps old versions of mappings along with the up-to-date versions. This way the evolution of a mapping can be recorded. M-Ontology mappings are then stored in a repository known as MO-Repository. Note that this approach by Li et al. does not propose a way to detect invalid mapping or repair them, it assumes that these actions are done elsewhere and the M-Ontology mappings are updated accordingly.

If a new concept from a dataset is to be added to the global query form, a new mapping has to be created between the global query form and the local query form of that dataset for that new concept. Instead of creating this mapping manually, an algorithm is proposed to reuse information within the existing mappings to aid in creating new ones. The algorithm works by using ontology matchers to discover semantically similar concepts from the mappings in the RO-Repository to the new concept to be added. The algorithm then examines the types of nodes (E, G or A) of similar concepts found and whether the node is part of an up-to-date mapping or an older version of a mapping. The algorithm will choose nodes from up-to-date mappings over older versions and will choose E nodes over G node and G nodes over A nodes. The intuition is that newer mappings are more desirable over new old ones and the lowest level of subclass as these represent finer gain concepts. When the algorithm has chosen the node, it retrieves the query reformulation code for that node. This code is then presented to a human user to verify or possibly modify to ensure it will work correctly. When verified by a human user, this code will now become a new node – defining a concept in a mapping.

The approach by Li et al. supports the mapping reuse use case:

- Discover information from existing mappings to aid in the creation of new alignments.

3.2.14. Alignment API

The Alignment API [Da11] is a java-based API designed to supplement applications with invoking ontology matching tools, representing alignments in a uniform way, storage of alignments and discovery of existing alignments. The Alignment API provides a language for representing alignments called Expressive and Declarative Ontology Alignment Language (EDOAL) (see Section 3.3.1). The API provides
functionality to store the alignments as EDOAL and also search for and retrieve existing alignments. Alignments can be searched based on source and target resources and the relationship between resources. Alignment can then be retrieved in EDOAL and used in other applications which also support the use of EDOAL.

Alignment API supports the mapping reuse use case:

- Discover alignments for sharing.

### 3.2.15. Sabou et al.

Sabou et al. propose an ontology matching approach that makes use of existing sets of ontologies - which include mappings between ontologies, as background, auxiliary information in the matching process. While other ontology matching approaches make use of other auxiliary information such as WordNet, Sabou et al. argue that the formally specified knowledge within ontologies would be less faulty that of less formally structured information sources. This approach makes use of the swoogle [Di04] semantic web search engine to search large amounts of existing ontologies and their mappings. The approach proposes a way to dynamically select relevant sets of existing ontologies and mappings that can be used as background information during the ontology matching process.

The approach works by checking to see if the source and target resources (that are involved in the matching process) exist in any other ontologies. The approach extends swoogle’s search capabilities through implementing string similarity techniques (common in other matching tools). The approach will initially try to discover mappings using the least number of ontologies. If mappings can be discovered between source and target resources using two ontologies, then the approach will stop searching. If it cannot discover mappings between source and target resources it will begin searching for mappings in multiple ontologies to where it then use the ‘mapping paths’ technique (see Section 3.2.8 for a description of mapping paths).

The approach by Sabou et al. supports the mapping reuse use case:

- Discover information from existing mappings to aid in the creation of new alignments.
- Discover mapping paths to aid in the creation of new mappings.
3.2.16. Section Summary

In summary, this section has presented fifteen existing approaches from the state of the art that support mapping maintenance, mapping reuse or both. As can be seen, the existing approaches vary in many ways: such as the mapping maintenance or mapping reuse features they support; whether the approach is explicitly designed for mapping maintenance or reuse; whether it is an ontology matching tool with mapping maintenance and reuse features; whether it performs maintenance or reuse over alignments or mappings. Table 3-1 below provides more insight into each approach.

Table 3-1. Breakdown of existing approaches as to whether they support performing maintenance and reuse over mappings or alignments, mappings or alignments derived from complex correspondences and whether they are domain independent or not.

<table>
<thead>
<tr>
<th>Existing Approach</th>
<th>Mapping or Alignment Supported</th>
<th>Support for Simple or Complex Correspondences</th>
<th>Domain Dependant or Independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>DyKOSMap</td>
<td>Alignment</td>
<td>No</td>
<td>Independent</td>
</tr>
<tr>
<td>DSNotify</td>
<td>Mapping (Interlink category)</td>
<td>No</td>
<td>Independent</td>
</tr>
<tr>
<td>Khattak et al.</td>
<td>Alignment</td>
<td>No</td>
<td>Independent</td>
</tr>
<tr>
<td>Maverick</td>
<td>Mapping (Vocabulary transformation category)</td>
<td>Yes</td>
<td>Independent (but will need to train with particular domain knowledge)</td>
</tr>
<tr>
<td>LogMap</td>
<td>Alignment</td>
<td>No</td>
<td>Independent</td>
</tr>
<tr>
<td>Dependency Model</td>
<td>Alignment</td>
<td>No</td>
<td>Independent</td>
</tr>
<tr>
<td>WOD-LMP</td>
<td>Mapping (Interlink category)</td>
<td>No</td>
<td>Independent</td>
</tr>
<tr>
<td>COMA++</td>
<td>Alignment</td>
<td>Yes</td>
<td>Independent</td>
</tr>
<tr>
<td>MooM</td>
<td>Alignment</td>
<td>Yes</td>
<td>Independent</td>
</tr>
<tr>
<td>LinkLion</td>
<td>Mapping (Interlink category)</td>
<td>No</td>
<td>Independent</td>
</tr>
<tr>
<td>BioPortal</td>
<td>Mapping (Interlink category)</td>
<td>No</td>
<td>Independent</td>
</tr>
<tr>
<td>Community Driven Ontology Mapping</td>
<td>Alignment</td>
<td>Yes</td>
<td>Independent</td>
</tr>
<tr>
<td>Li et al.</td>
<td>Mapping (Vocabulary transformation category)</td>
<td>Yes</td>
<td>Independent</td>
</tr>
<tr>
<td>Alignment API</td>
<td>Alignments</td>
<td>Yes</td>
<td>Independent</td>
</tr>
<tr>
<td>Sabou et al.</td>
<td>Mapping (Interlink category)</td>
<td>No</td>
<td>Independent</td>
</tr>
</tbody>
</table>
Specifically the table displays whether an approach supports performing maintenance and reuse over mappings or alignments, whether it supports alignments and mappings derived from complex correspondences and whether the approach is domain dependant (i.e. will only work on data from the medical domain for example) or domain independent.

3.3. Existing Approaches for Ontology Mapping Representation

As seen from Section 3.2, there are a number of existing approaches for performing maintenance and reuse over alignments. The research in this thesis is focused on performing maintenance and reuse directly over mappings between Linked Data datasets. Therefore a mapping representation had to be selected to support the practical undertaking of the research.

This section reviews ten existing approaches for representing ontology mappings. In the view of the author, this is an exhaustive list of existing approaches to representing Linked Data ontology mappings. Recall from Chapter 1, the research in this thesis is not focused on performing mapping maintenance and reuse over uplift and lowering category mappings, therefore representations for these two categories are not considered in this review. The characteristics of the approaches that were chosen for examination were: what category a mapping is designed to represent, whether the approach provides and supports annotation with meta-data (as meta-data has been shown to be useful in mapping reuse [Th13]), how the representation is encoded (useful for how mappings will be stored and accessed) and (only applicable to vocabulary transformation category of mappings) if the representation is directly executable. As the vocabulary transformation category of mappings are used to transform instance data that is represented according to the vocabulary of one dataset into the vocabulary of another dataset, having a mapping representation that is executable to perform this transformation is useful and desirable.

3.3.1. LinkLion Mapping

As stated in Section 3.2.10, LinkLion provides a mapping representation for representing interlink category mappings. Figure 3-3 presents an example of a LinkLion mapping representing an interlink category mapping between the DBpedia and Linked Movie Database datasets. Mappings are modelled as instances of the llont:Mapping class. Meta-data terms are annotated to the mapping instance. Individual interlinks are
modelled as an instance of a `llont:Link` class. The `rdf:subject`, `rdf:predicate` and `rdf:object` properties are used to model the subject, predicate and object of the interlink respectively. The interlinks are linked to a mapping instance via the `prov:wasDerivedFrom` property.

The LinkLion mapping is a fine grain mapping representation, encoded in RDF for representing interlink category mappings between datasets, therefore they are not applicable for execution.

```xml
<http://www.linklion.org/mapping/053e390a04458cc8c296388b2e39db3c> a llont:Mapping;
  prov:generatedAtTime "2014-03-17T10:31:07+01:00";
  prov:wasGeneratedBy llaig:limesDefault;
  llont:hasSource lldat:dbpedia.org;
  llont:hasTarget lldat:data.linkedmdb.org;
  prov:wasDerivedFrom llmap:053e390a04458cc8c296388b2e39db3c.

<http://www.linklion.org/link/0504622dab90daaf6af277ff7df26c83> a llont:Link;
  prov:wasDerivedFrom llmap:053e390a04458cc8c296388b2e39db3c;
  rdf:subject <http://dbpedia.org/resource/Richard_III_%281995_film%29>;
  rdf:predicate owl:sameAs;
```

Figure 3-3. LinkLion mapping representing an interlink category mapping between the DBpedia and Linked Movie Database datasets.

### 3.3.2. BioPortal Mapping

BioPortal provides a mapping representation for representing interlink category mappings as stated in Section 3.2.11. A BioPortal mapping is modelled as an instance of the `map:One_To_One_Mapping` class. Meta-data terms are annotated to the mapping instance. The `map:source`, `map:relation` and `map:target` properties are used to model the subject, predicate and object of the interlink respectively. Figure 3-4 presents an example of a BioPortal mapping between two biomedical databases that are stored in the BioPortal repository.

Similar to the LinkLion mapping, the BioPortal mapping is a fine grain mapping representation, encoded in RDF for representing interlink category mappings between datasets and are not applicable for execution.
Figure 3-4. BioPortal Mapping representing an interlink category mapping between two biomedical databases in the BioPortal repository.

3.3.3. OWL
The Web Ontology Language (OWL) [Mo09] is a general purpose ontology language that contains a number of classes and properties that are used to model ontologies and vocabularies. OWL is not a mapping representation language but does provide support to represent some basic mappings. OWL allows constraints to be placed on resources in ontologies and relationships to be declared on resources between two or more ontologies, such as sameAs, equivalentClass, unionOf, intersectionOf, equivalentProperty and inverseOf. These constraints and relationships are stored as part of the ontologies themselves and based on these constraints and relationships, additional information about instances can be inferred through the use of an OWL reasoner. This allows OWL to represent interlinks between datasets. However OWL is not expressive enough to represent the vocabulary transformation category of mappings as it does not natively provide support for property value transformations (such as numeric or string transformations).

3.3.4. Jena-Rules
Jena-Rules [Ca04, Ap16] are part of the Apache Jena toolkit, which provides an inference engine to perform reasoning over RDF data using rules. The rules are written in the Datalog programming language and are built on top of RDF datasets. Jena-Rules can be used to infer new information about instances, supports the transformation of RDF data from one vocabulary to another and can impose constraints on data (such as a percentage value cannot be less than 0 or greater than 100, or a date must be in the
format dd/mm/yyyy). Jena-Rules can be used to represent both the interlink category and vocabulary transformation category of mappings. However its expressivity is limited due to little support for string transformation functions. Jena-Rules are executable by a Jena Reasoner.

3.3.5. SWRL
The Semantic Web Rule Language (SWRL) [Ho04] is for establishing rules on RDF data. SWRL is a combination of OWL-Description Logic (OWL-DL) [Mc04], which is a more expressive version of OWL and the Rule Markup Language (RuleML) [Ru16]. Similar to Jena-Rules SWRL can be used to infer new information about instances, supports the transformation of RDF data from one vocabulary to another and can impose constraints on data. SWRL supports a number of complex string functions, making it more expressive than Jena-Rules. This makes SWRL suitable for representing both the interlink category and vocabulary transformation category of mappings. Jena-rules execution is only supported by certain OWL reasoners such as Protégé, Pellet and Hermet

3.3.6. SPARQL CONSTRUCT Queries
The SPARQL Protocol and RDF Query Language (SPARQL) [Ha13a] is the W3C recommendation query language for RDF data. SPARQL CONSTRUCT queries have been proposed [Eu08] to represent executable vocabulary transformation categories of mappings between Linked Data datasets that are executed on a standard SPARQL processor. CONSTRUCT queries can convert data represented in one vocabulary to another vocabulary and supports a wide range of functions (such as numeric functions, string transformation functions, converting strings to a URI and a URI to a string) allowing it to perform complex data transformations. CONSTRUCT queries can also be used to represent single triples. This makes SPARQL CONSTRUCT queries useful for representing both the vocabulary transformation and interlink category of mappings.

3.3.7. SPIN Rules
SPARQL Inference Notation (SPIN) Rules [Kn11] are rules that are based on SPARQL queries. They contain all the functionality of SPARQL queries but they can be extended to include custom functions and imposes constraints on data. While based on SPARQL queries, they are not the same thing. For example SPIN rules are encoded in RDF -
described using the SPIN SPARQL syntax. SPIN rules cannot be executed by a SPARQL processor; they are executed by a SPIN inference engine.

3.3.8. R2R Mapping Language

The R2R Mapping Language [Bi10] is an RDF-based mapping language designed to represent vocabulary transformation category of mappings and are executed by the R2R Framework. Mappings are modelled as an instance of the r2r:Mapping class. Meta-data terms are annotated to the mapping instance. The r2r:sourcePattern and r2r:targetPattern properties are used to represent triple patterns which indicate how data is transformed from source to target. These triple patterns are represented as strings. The r2r:transformation property is used to represent any data transformations involved in the mapping, also as a string. Similar to SPARQL construct queries, these mappings are suitable to represent mappings between Linked Data datasets, as they can convert data represented in one vocabulary to another and support a wide variety of data transformations. Figure 3-5 presents an example of a R2R Mapping. Due to the source and target triple patterns and transformations represented as strings, the R2R Mapping language is not a fine grain representation.

```xml
mappings:dbpediaToLinkedmdbRuntime a r2r:Mapping;
  r2r:sourcePattern "?SUBJ <http://dbpedia.org/ontology/runtime> ?runtimeInSeconds";
  r2r:targetPattern "?SUBJ <http://data.linkedmdb.org/resource/movie/runtime> ?runtimeInMinutes";
  r2r:transformation "?runtimeInMinutes = integer(?runtimeInSeconds / 60)";
  dc:creator <http://www4.wiwiss.fu-berlin.de/is-group/resource/persons/Person30>;
  dc:date "2010-06-22"^^xsd:date .
```

Figure 3-5. R2R mapping for a movie runtime vocabulary resource between the DBpedia and Linked Movie Database datasets.

3.3.9. Alignment Format and EDOAL

The Alignment Format [Eu04] is designed to represent alignments between datasets. It is concerned with representing non-complex one-to-one alignments - encoded in RDF. It simply represents a source and target dataset resource and the correspondence between them, using one of the following relations: > (subsumes), < (is subsumed), = (equivalent), % (incompatible), HasInstance, InstanceOf. The Expressive and
Declarative Ontology Alignment Language (EDOAL) [Da11] was introduced as an extension to the Alignment Format. EDOAL is also designed to represent alignments between datasets but can represent more complex, many-to-many alignments. EDOAL also allows for more expressive correspondences to be declare than the Alignment Format using the operators, and, or, not combined with equivalent, subsumption, inverse and transitive in addition to those of the Alignment format.

While the Alignment format and EDOAL do not represent mappings, it is included here as it is a popular format for representing alignments.

3.4. Mapping Maintenance and Reuse Use Cases and Tasks

This section presents the mapping maintenance and reuse use cases that were identified during the review of existing approaches, examined in Section 3.2. Each of these use cases are then examined to determine the individual tasks that are needed to perform the use case. In this thesis, SUMMR templates are evaluated based on their performance of these tasks. The rational for deriving these tasks and evaluating the templates around them is that by showing the flexibility of the templates. If the templates can perform the tasks, then these tasks can be combined in order to perform the use cases. The additional envisaged benefit of this flexibility is that tasks can be combined together to perform new mapping maintenance and reuse use cases that may appear in the future.

A total of 11 different use cases were identified from reviewing the existing approaches. However, some of these use cases are very similar, such as the following three:

- Discover invalid alignment due to changes in datasets an alignment references
- Discover invalid interlinks due to changes in datasets an interlink references.
- Discover invalid mappings due to changes in datasets a mapping references.

The difference between the three is what is being checked to be invalid - an alignment, an interlink or a mapping. Not considering the similar use cases and using the term “mapping” in the use cases to replace the terms “alignment” and “interlink”, six use cases were identified. Table 3-2 displays these six use cases and categorises them according to their mapping reuse or maintenance activities.
Table 3-2. Six mapping maintenance and reuse use cases.

<table>
<thead>
<tr>
<th>Use Case ID No.</th>
<th>Use Case</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Retrieve mappings for sharing</td>
<td>Reuse</td>
</tr>
<tr>
<td>2</td>
<td>Discover information from existing mappings to aid in the creation of new mappings.</td>
<td>Reuse</td>
</tr>
<tr>
<td>3</td>
<td>Discover mapping paths to aid in the creation of new mappings</td>
<td>Reuse</td>
</tr>
<tr>
<td>4</td>
<td>Discover back links to aid in the creation of new mappings.</td>
<td>Reuse</td>
</tr>
<tr>
<td>5</td>
<td>Discover invalid mappings due to changes in datasets a mapping references.</td>
<td>Maintenance</td>
</tr>
<tr>
<td>6</td>
<td>Repair of an invalid mapping.</td>
<td>Maintenance</td>
</tr>
</tbody>
</table>

By examining the six use cases, the author identified five tasks that need to be supported in order to perform the use cases. In the following use case descriptions, the supporting tasks are referenced using bullet points. A summary of the tasks are then presented after.

Use Case 1 - Retrieve Mappings for Sharing

This use case involves finding particular mappings that meet specific criteria so they can be shared and used in a situation that requires mappings meeting that criteria. The tasks needed to perform this use case are:

- **Discover a mapping based on search criteria.** Relevant mappings that adhere to search criteria must be discovered.
- **Retrieve the entirety of a mapping.** When relevant mappings have been discovered, the entirety of the mappings must be retrieved so they can be shared.

Use Case 2 - Discover Information from Existing Mappings to Aid in the Creation of New Mappings

In this use case, information within existing mappings are discovered so that information can help with a decision when creating a new mapping. This can be information discovered by automated systems to help with an ontology matching task or information discovered by human which will provide insight on certain mappings which will allow the user to create similar mappings more easily. The tasks needed to perform this use case are:

- **Discover a mapping based on search criteria.** Relevant mappings of specific search criteria must be discovered.
• **Discover and retrieve individual properties of a mapping.** Once relevant mappings have been discovered, additional information, inherent within the properties from these mappings can then be discovered and retrieved. This information may be meta-data about a mapping, specific source or target resources referenced by a mapping or a property value transformation than the mapping contains.

**Use Case 3 - Discover Mapping Paths to Aid in the Creation of new Mappings**

This use case involves discovering mapping paths within existing mappings. The results from the mapping paths (which will be a set of source and target resources) can be used to create new mappings. The tasks needed to perform this use case are:

• **Discover a mapping based on search criteria.** First, a set of relevant mappings must be discovered - for example mappings that contain a resource from Dataset-A as the source of the mapping and Dataset-B as the target of the mapping. Second, another set of relevant mappings must be discovered - for example mappings that contain a resource from Dataset-B as the source and Dataset-C as the target. Mappings from both sets where the Dataset-B resources are the same should only be discovered.

• **Discover and retrieve individual properties of a mapping.** Next step is to discover and retrieve the source resources from the first set of mappings and the target resources from the second set of mappings. These retrieved source and target resources can then be used in the creation of new mappings.

**Use Case 4 - Discover Back Links to Aid in the Creation of new Mappings:**

This use case is for interlink category mappings only and involves discovering a set of interlinks and switching the position of the subject and object instances of the interlinks. The tasks needed to perform this use case are:

• **Discover a mapping based on search criteria.** First, a set of relevant mappings must be discovered between a source and target dataset.

• **Discover and retrieve individual properties of a mapping.** Next step is to discover the source and target properties of the discovered interlink category mappings and retrieve those properties so they are switched around. The switched properties will be new source and target instances that can be used to create new interlink category mappings.
Use Case 5 - Discover Invalid Mappings due to Changes in Datasets

This use case is concerned with discovering mappings that would be classed as invalid due to changes in datasets that the mappings reference. This involves checking to see if the source and target resources of a mapping are still valid in relation to the datasets those resources reference. The tasks needed to perform this use case are:

- **Discover a mapping based on search criteria.** First, a set of relevant mappings must be discovered between the source and target datasets.

- **Discover and retrieve individual properties of a mapping.** Next, the source and target resources of the relevant mappings need to be discovered and retrieved.

- **Compare mapping properties to resources from a dataset.** The retrieved source and target mapping resources need to be compared to resources from a source and target dataset respectively. Accessing a datasets resources can be in the form of directly access the dataset, accessing the resources that are recorded in a change log file, or accessing a machine learned representation of those resources. The comparison of the mapping properties to dataset resources must detect a change between the two. When this change is detected, the mapping is flagged as invalid.

Use Case 6 - Repair of an Invalid Mapping

As stated previously, an invalid mapping is a mapping that no longer produces correct results due to changes in the representation of vocabulary terms or instances in the datasets referenced by the mapping. Therefore in order to repair the mapping and have it produce correct results, the changed vocabulary terms or instances need to be replaced with the updated vocabulary terms and instances from the datasets the mapping references. Therefore to repair a mapping, the invalid vocabulary terms or instances have to be known and the updated terms or instances (to replace the invalid ones) also have to be known. Once these are known, the invalid terms or instances need to be deleted and replaced with the updated terms or instances. This could be done manually, but with large numbers of mappings it would be a time consuming task. The challenge with the automatic repair of mappings is to automatically discover the invalid vocabulary terms and instances, automatically discover valid (updated) vocabulary terms and instances and then automatically alter a mapping through deleting the invalid ones and replacing them with the valid.
This use case is concerned with altering an invalid mapping, as to replace invalid mapping properties, with valid properties in order for the mapping to be valid. It is independent as to whether this is done through manual or automatic means. The tasks needed to perform this use case are:

- **Alteration of an Existing Mapping.** A known invalid mapping must have its invalid properties deleted and replaced with updated valid properties. This will restore validity to the mapping.

In summary, the five tasks that have been identified by the author, are presented in Table 3-3.

<table>
<thead>
<tr>
<th>Task ID No.</th>
<th>Task Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Discover a mapping based on search criteria</td>
</tr>
<tr>
<td>2</td>
<td>Discover and retrieve individual properties of a mapping</td>
</tr>
<tr>
<td>3</td>
<td>Retrieve the entirety of a mapping</td>
</tr>
<tr>
<td>4</td>
<td>Compare mapping properties to resources from a dataset</td>
</tr>
<tr>
<td>5</td>
<td>Alteration of an Existing Mapping</td>
</tr>
</tbody>
</table>

It is argued by the author of this thesis that any approach seeking to support both mapping maintenance and reuse should support these tasks.

### 3.5. Analysis of Existing Approaches for Mapping Maintenance and Reuse and for Mapping Representation

In this section it is shown to what extent the use cases and tasks, described in Section 3.4, are supported by the existing approaches that were reviewed in Section 3.2. In addition, a comparison of the characteristics of the mapping representations presented in Section 3.3 is provided.

*Table 3-4* presents the extent to which each approach supports use cases and tasks for mapping maintenance and reuse. It was found that none are explicitly designed to support both mapping maintenance and reuse. None of the approaches support all six of the mapping maintenance and reuse use cases identified and therefore, none of them support all five of the mapping maintenance and reuse tasks identified.
Table 3-4. Mapping maintenance and reuse use cases and tasks supported by the existing approaches reviewed from the state of the art.

<table>
<thead>
<tr>
<th>Existing Approach</th>
<th>Use Cases Supported</th>
<th>Tasks Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>DyKOSMap</td>
<td>X X X X X X</td>
<td></td>
</tr>
<tr>
<td>DSNotify</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Khattak et al.</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>Maveric</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>LogMap</td>
<td>X X X X X</td>
<td></td>
</tr>
<tr>
<td>Dependency Model</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>WOD-LMP</td>
<td>X X X X X</td>
<td></td>
</tr>
<tr>
<td>COMA++</td>
<td>X X X X X X</td>
<td></td>
</tr>
<tr>
<td>MooM</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>LinkLion</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>BioPortal</td>
<td>X X X X</td>
<td></td>
</tr>
<tr>
<td>Community Driven Ontology Mapping</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Li et al.</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Alignment API</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Sabou et al.</td>
<td>X X X</td>
<td></td>
</tr>
</tbody>
</table>

The most supported use case is number 5 - Discover invalid mappings due to changes in datasets a mapping references. Out of the seven approaches that support this use case, only two additionally support number 6 - Repair of an invalid mapping. The least supported use case is number 2 - Discover information from existing mappings to aid in the creation of new mappings. The two approaches that support this use case also do not support any other use cases. The most supported tasks is number 1 - Discover a mapping based on search criteria, which is supported by all existing approaches. Any approach that is to perform mapping maintenance or reuse must be able to discover the mappings first. The least supported task is number 6 - Alteration of an Existing Mapping.

Table 3-5 compares the key characteristics of the mapping representations reviewed. From the review of the eight existing approaches for mapping representation, seven can represent the interlink category of mappings, five can represent the vocabulary transformation category of mappings and four can represent both categories. Four of the mappings representations provide and support annotation with mapping meta-data. Five mapping representations are encoded in the RDF standard, Jena-Rules and SWRL are encoded in specific rule language syntax and SPARQL CONSTRUCT queries are encoded in SPARQL query syntax. Three of the mapping representations operate through inference/reasoning and two can only be directly executed by an application.
Table 3-5. Characteristics of Existing Mapping Representations Reviewed.

<table>
<thead>
<tr>
<th>Existing Mapping Representation</th>
<th>Category of Mapping Represented</th>
<th>Mapping Meta-data</th>
<th>Encoding</th>
<th>Means of Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinkLion Mapping</td>
<td>Interlink</td>
<td>Yes</td>
<td>RDF</td>
<td>N/A</td>
</tr>
<tr>
<td>BioPortal Mapping</td>
<td>Interlink</td>
<td>Yes</td>
<td>RDF</td>
<td>N/A</td>
</tr>
<tr>
<td>OWL</td>
<td>Interlink</td>
<td>No</td>
<td>OWL/RDF</td>
<td>N/A</td>
</tr>
<tr>
<td>Jena-Rules</td>
<td>Interlink</td>
<td>No</td>
<td>Datalog</td>
<td>Jena Reasoner</td>
</tr>
<tr>
<td></td>
<td>Interlink</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWRL</td>
<td>Interlink</td>
<td>No</td>
<td>RuleML</td>
<td>Certain OWL Reasoners</td>
</tr>
<tr>
<td></td>
<td>Vocabulary Transformation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPARQL CONSTRUCT</td>
<td>Interlink</td>
<td>No</td>
<td>SPARQL Query</td>
<td>SPARQL Processor</td>
</tr>
<tr>
<td></td>
<td>Vocabulary Transformation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPIN Rules</td>
<td>Interlink</td>
<td>No</td>
<td>RDF</td>
<td>SPIN Inference Engine</td>
</tr>
<tr>
<td></td>
<td>Vocabulary Transformation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R2R Mapping Language</td>
<td>Vocabulary Transformation</td>
<td>Yes</td>
<td>RDF</td>
<td>R2R Framework</td>
</tr>
<tr>
<td>Alignment Format and EDOAL</td>
<td>Alignment</td>
<td>Yes</td>
<td>RDF</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Of the mapping representations examined, three in particular stand out. These are the LinkLion Mapping, the BioPortal Mapping and the R2R Mapping Language. These three are dedicated representations, providing their own vocabulary to model and describe them (as seen in the examples in Section 3.3). Mappings modelled and described in such a way should make it easier to discover and retrieve specific mappings and properties within those mappings – which is appealing for mapping maintenance and reuse tasks. These mapping representations also provide and support mapping meta-data and are encoded in RDF which means they can be stored alongside Linked Data datasets. The R2R Mapping Language is executable, which means it can transform data represented in one vocabulary into another vocabulary. The LinkLion Mapping and the BioPortal Mapping are not executable as they represent interlink category mappings only. The drawback of these three representations is that they do not represent both vocabulary transformation mappings and interlink categories of mappings.

The review of the existing approaches for the mapping representation has provided information useful to the decision making related to which representation to use during
the design of the SUMMR methodology, specifically whether to choose an existing mapping representation or design a new one.

3.6. Chapter Summary

This chapter has presented a state of the art review on existing approaches that support mapping maintenance, mapping reuse or both. This review was performed to identify the mapping maintenance and reuse use cases that are supported in the state of the art, as no taxonomy exists as of yet. Six use cases were identified. Five individual tasks that are needed to support the use cases were derived by the author of this thesis. The author of this thesis argues that these tasks must be supported by any approach for both mapping maintenance and reuse. A gap in the state of the art was found as none of the existing approaches support all six of the use case or all five of the tasks. Identifying the mapping maintenance and reuse use cases and tasks has provided information with regards to designing a new approach that will support all the mapping maintenance and reuse use cases and tasks.

In this chapter, a state of the art review on existing approaches for vocabulary transformation and interlink category of mapping representation was also undertaken. The review was performed to determine the characteristics of the existing mapping representations. These characteristics will be considered when choosing a mapping representation, or designing a new one to perform maintenance and reuse over for the mapping maintenance and reuse approach designed in this thesis.

Building on the review and analysis of the state of the art, the next chapter refines the six use cases identified in this chapter. These refined use cases detail how the six mapping maintenance and reuse use cases, should be performed by a query-based methodology for mapping maintenance and reuse.
4. Refinement of Use Cases for a Query-Based Methodology for Mapping Maintenance and Reuse

This is a brief chapter which presents six refined use cases which describe how a query-based methodology would perform the six mapping maintenance and reuse use cases identified in Chapter 3. Section 4.1 presents the refined use cases. Section 4.2 concludes the chapter with a summary.

4.1. Refined Use Cases

This section presents six refined use cases, which provides a detailed breakdown of what is required to perform the six mapping maintenance and reuse use cases when using a query-based methodology. The six refined use cases are presented in UML. For each refined use case, a description of the preconditions, the actors involved and what is involved in the query to perform the use case is given. Also, within each use case it is highlighted where a mapping maintenance and reuse task is involved.

4.1.1. Refined Use Case 1: Retrieve Specific Mappings for Sharing

Refined use case 1 is described below:

- **Name:** Retrieve specific mappings for sharing.
- **Description:** A user wishes to retrieve specific mappings from the database for sharing purposes. This use case is depicted in Figure 4-1.
- **Assumptions:**
  - A number of mappings already exist within the database.
  - The database has a built in query processor.
- **Actors:**
  - User.
  - Database.
- **Steps:**
  1. User decides to share specific mappings.
  2. User creates a query, specifying the specific criteria of the mappings to be retrieved. **Task 1: Discover a mapping based on search criteria. Task 3: Retrieve the entirety of a mapping.**
  3. User executes the query, via the query processor, over the mappings.
  4. Database returns mappings to the user.
4.1.2. Refined Use Case 2: Discover Information from Existing Mappings to Aid in the Creation of New Mappings

Refined use case 2 is described below:

- **Name:** Discover information from existing mappings to aid in the creation of new mappings.

- **Description:** A user wishes to discover if any mappings in the database contain information that can aid in the creation of a new mapping. Consider the following example: a user wishes to create a new mapping using a movie’s *duration* from the DBpedia dataset as the mapping’s source to a movie’s *duration* in the FreeBase dataset as the mapping’s target. The user does not know the correct vocabulary term to use for a movie’s duration from the FreeBase dataset. The user also knows that a movie’s duration is represented in seconds in the DBpedia dataset but unsure how it is represented in the FreeBase dataset. So the user wishes to find any existing mappings to a movie’s *duration* in the FreeBase dataset in order to find the correct FreeBase vocabulary term to use and how to represent the duration of a movie – whether it be seconds or minutes or hours. This use case is depicted in Figure 4-2.

Figure 4-1. UML diagram for the refined use case of retrieving specific mappings for sharing.
Figure 4-2. UML diagram for the refined use case of discovering information from existing mappings to aid in the creation of new mappings.

- **Assumptions:**
  - A number of mappings already exist within the database.
  - The database has a built-in query processor.
  - The user is aware of the mappings to create between two datasets but unsure of the exact details to use.

- **Actors:**
  - User.
  - Database.

- **Steps:**
  1. User decides to create a new mapping (for example – between a movie’s `duration` from the DBpedia dataset as source to a movie’s `duration` in the FreeBase dataset as the target).
  2. User creates a query to:
     a. Discover mappings based on the specific search criteria of the mappings to be retrieved (for example, mappings that contain the
words “freebase” and “duration” in the target property of the mapping). **Task 1: Discover a mapping based on search criteria.**

b. Then based on the mappings discovered in step 2(a), discover and retrieve all the properties of each mapping. **Task 2: Discover and retrieve individual properties of a mapping.**

3. User executes the query, via query processor, over the mappings in the database.
4. Database returns mappings to the user.
5. User then examines the returned mapping properties (which will contain the FreeBase vocabulary terms used and indicates how a movie’s duration is represented).

### 4.1.3. Refined Use Case 3: Discover Mapping Paths to Aid in the Creation of New Mappings

Refined use case 3 is described below:

- **Name:** Discover mapping paths to aid in the creation of new mappings.
- **Description:** A user wishes to discover mapping paths amongst the mappings in the database for the purposes of aiding in the creation of new mappings.
- **Assumptions:**
  - A number of mappings already exist within the database.
  - The database has a built in query processor.
  - The user is aware of the mappings that exist between different datasets in the database.

- **Actors:**
  - User.
  - Database.
Steps:

1. User decides to discover mapping paths to aid in the creation of new mappings between two datasets DS-A and DS-C.

2. User creates a query to:
   a. Discover mappings that have a DS-A resource as the source property of the mapping, and resources from an intermediate dataset - DS-I, as the mapping target. Task 1: Discover a mapping based on search criteria.

   b. Then discover mappings that use the resources from DS-I (from step 2(a)) as the source property of the mapping, and have resources from DS-C as the target property of the mapping. Task 1: Discover a mapping based on search criteria.

   c. Return the source property of the mappings discovered in step 2(a) (which will be the source of the new mappings) and the target property of the mappings discovered in step 2(b) (which will be the
target of the new mappings). **Task 2: Discover and retrieve individual properties of a mapping.**

3. User executes the query, via the query processor, over the mappings in the database.

4. Database returns results to the user.

### 4.1.4. Refined Use Case 4: Discover Back Links to Aid in the Creation of New Mappings

Refined use case 4 is described below:

- **Name:** Discover back links to aid in the creation of new mappings
- **Description:** A user wishes to discover interlink category mappings between two datasets, DS-A as the *source*, and DS-B as the *target*, so the *source* and *target* can be reversed to make new interlink category mappings between DS-B as the *source* and DS-A as the *target*. This use case is depicted in Figure 4-4.
- **Assumptions:**
  - A number of mappings already exist within the database.
  - The database has a built-in query processor.
- **Actors:**
  - User.
  - Database.
- **Steps:**
  1. User decides to create new interlink category mappings between two datasets (DS-B as the *source*, and DS-A as the *target*).
  2. User creates a query to:
     a. Discover mappings that have a DS-A instance as the *source* property of the interlink and a DS-B instance as the *target* property of the interlink. **Task 1: Discover a mapping based on search criteria.**
     b. Return the *source* property of the mappings (which will be the *target* of the new interlink category mappings) and return the *target* property of the mappings (which will be the *source* of the new interlink category mappings). **Task 2: Discover and retrieve individual properties of a mapping.**
  3. User executes the query, via the query processor, over the mappings.
  4. Database returns results to the user.
4.1.5. Refined Use Case 5: Discover Invalid Mappings due to Changes in Datasets a Mapping References

Refined use case 5 is described below:

- **Name:** Discover invalid mappings due to changes in datasets a mapping references.
- **Description:** A user wishes to discover if mappings have become invalid due to changes in the datasets the mappings reference. This use case is depicted in Figure 4-5.
- **Assumptions:**
  - A number of mappings already exist within the database.
  - The database has a built in query processor.
  - The user has access to the datasets the mappings to be checked reference (this can be a dump file of a dataset or online access to the dataset through a federated call).
Discover Invalid Mappings due to Changes in Datasets a Mapping References

Figure 4-5. UML diagram for the refined use case of discovering invalid mappings due to changes in datasets a mapping references.

- **Actors:**
  - User.
  - Database.

- **Steps:**
  1. User decides to check if mappings, that reference two specific datasets, DS-A and DS-B, have become invalid due to changes within those datasets.
  2. **IF** DS-A is to be accessed through a dump file.
     THEN the user loads the DS-A dump file into the database.
     ELSE move on to step 3.
  3. **IF** DS-B is to be accessed through a dump file.
     THEN the user loads the DS-B dump file into the database.
     ELSE move on to step 4.
  4. User creates a query to:
a. Discover mappings that have a DS-A resource as the source property of the mapping and a DS-B resource as the target property of the mapping. **Task 1: Discover a mapping based on search criteria.**

b. **IF** DS-A is accessed through a dump file.

**THEN** Compare the source property of the mappings to the resources of DS-A to see if the source properties exist. **Task 2: Discover and retrieve individual properties of a mapping. Task 4: Compare mapping properties to resources from a dataset.**

**ELSE IF** DS-A is accessed through a federated call.

**THEN** Compare the source property of the mappings to the resources of DS-A to see if the source properties exist. **Task 2: Discover and retrieve individual properties of a mapping. Task 4: Compare mapping properties to resources from a dataset.**

c. **IF** DS-B is accessed through a dump file.

**THEN** Compare the target property of the mappings to the resources of DS-B to see if the target properties exist. **Task 2: Discover and retrieve individual properties of a mapping. Task 4: Compare mapping properties to resources from a dataset.**

**ELSE IF** DS-B is accessed through a federated call.

**THEN** Compare the target property of the mappings to the resources of DS-B to see if the target properties exist. **Task 2: Discover and retrieve individual properties of a mapping. Task 4: Compare mapping properties to resources from a dataset.**

d. Return the name of the mappings and its source and target properties (which will be mappings with a source and/or target property that does not exist in DS-A or DS-B respectively.

5. User executes the query, via the query processor over the mappings and the datasets in the database.

6. Database returns results to the user.

**4.1.6. Refined Use Case 6: Repair of an Invalid Mapping**

Refined use case 6 is described below:

- **Name:** Repair of an invalid mapping.
• **Description:** A user wishes to repair a mapping that was discovered to be invalid. This use case is depicted in *Figure 4-6.*

![Figure 4-6. UML diagram for the refined use case of the repair of an invalid mapping.](image)

• **Assumptions:**
  o The invalid mapping is in the database.
  o The database has a built-in query processor.
  o The user knows the name of the invalid mapping and the mapping’s *source* and *target* properties that are invalid (which can be obtained from *use case 5*).
  o The user has *new resources* from datasets (that the invalid mapping references) which will be used to replace the invalid resources in the invalid mapping.

• **Actors:**
  o User.
  o Database.

• **Steps:**
  1. User decides to repair a mapping previously discovered to be invalid.
  2. User creates a query where:
     a. The name of the invalid mapping is specified.
     b. The invalid properties of the mapping are removed through use of a DELETE clause. *Task 5: Alteration of an existing mapping.*
c. The *new resources* from datasets, which will be used to replace the invalid properties are inserted through use of an INSERT clause.

*Task 5: Alteration of an existing mapping*

3. User executes the query, via the query processor over the invalid mapping in the database to repair the mapping.

### 4.2. Chapter Summary

In summary, this chapter has presented a set of six refined use cases. These refined use cases are refinements of the six mapping maintenance and reuse use cases identified from the state of the art review carried out in *Chapter 3*. The refined use cases provide a detailed breakdown of what is required, when using a query-based methodology, to perform the use cases - indicating the preconditions, the actors involved and what a query must do.

The six refined use cases proved useful when it came to the designing the query-based methodology in the thesis (see next chapter - *Chapter 5*), particularly for the design of the query templates it provides.
5. Design of the SUMMR Methodology

This chapter describes the SPARQL Usage for Mapping Maintenance and Reuse (SUMMR) methodology. Section 5.1 introduces SUMMR. Section 5.2 presents the elements of SUMMR’s design. Section 5.3 presents how users would use the SUMMR methodology and where it would fit in a mapping lifecycle. Section 5.4 presents an implementation of SUMMR and briefly discusses other possible tools that would support the SUMMR approach. Reflections upon the applicability and limitations of the design of SUMMR are discussed in Section 5.5. To conclude the chapter, Section 5.6 presents a summary.

5.1. Introduction to the SUMMR Methodology

SUMMR is a methodology for the maintenance and reuse of mappings between Linked Data datasets. SUMMR provides a set of practices, through the elements of its design and a set of procedures and rules in the form of query templates for performing mapping maintenance and reuse. The aim of SUMMR is to provide a methodology, to support performing mapping maintenance and reuse. It is envisioned that it will be beneficial to such roles by reducing the time and effort for (i) the detection and repair of invalid mappings and (ii) the creation of new mappings, through the reuse of existing mappings. To achieve this, SUMMR specifies how the mappings and the datasets that mappings reference should be accessed. SUMMR provides query templates for the five mapping maintenance and reuse tasks and six use cases, which were identified from the state of the art (Chapter 3). Since different approaches exist for representing mappings, the templates will vary depending on the mapping representation being considered. Therefore, templates are specialised for different mapping representations. While it is not mandatory to use with SUMMR, SUMMR includes a mapping representation - named the SPARQL Centric Mapping Representation (SCMR) that is used to represent both vocabulary transformation and interlink categories of mappings.

Mapping maintenance and reuse involves searching for, retrieving and manipulating (potentially large numbers of) mappings and comparing mappings to datasets that they reference. Databases and query languages are technologies that SUMMR relies upon for this. Databases provide a way to readily organise data and a query language allows for the searching, retrieval, manipulation and comparison of that data. As Linked Data datasets are encoded in RDF, they are stored in triple-stores and accessed through the
SPARQL query language [Je11]. SPARQL is the W3C recommended query language for RDF data. Since SUMMR is designed for Linked Data mappings, it follows these same practices and relies on a triple-store to store RDF-based mappings and datasets, and also relies on SPARQL to perform mapping maintenance and reuse. The motivation behind relying upon a triple-store and SPARQL is that these two technologies are likely to be deployed by Linked Data dataset providers and maintainers. In addition, SPARQL is a standards-based approach for querying which highlights that SUMMR is a query-based methodology for mapping maintenance and reuse. As seen in the state of art (Chapter 3), some existing approaches also rely upon SPARQL for mapping maintenance and reuse, but to a limited degree (such as MooM which only uses SPARQL queries to search for mappings based on mapping meta-data fields). Other existing approaches rely upon ad-hoc methods and tools for maintenance and reuse. It is envisaged that, by relying upon SPARQL, it will encourage the adoption of the SUMMR methodology by Linked Data dataset providers and maintainers.

5.1.1. Brief Overview of SUMMR

SUMMR provides SPARQL templates that can be used to undertake mapping maintenance and reuse tasks and use cases, and that can be specialised to different mapping representations. SUMMR provides its own mapping representation (SCMR). In this brief overview of SUMMR, it is assumed that mappings will be represented in SCMR and templates are specialised for SCMR, see Figure 5-1 below.

First, mappings need to be converted to SCMR. This could be done through a software implementation to transform, for example SPARQL CONSTRUCT queries into SCMR. One such possible implementation is discussed further in Section 5.4.2. In the current design, the SCMRs are stored in a triple store. Datasets that mappings reference may need to be accessed in order to discover if any of the mappings are invalid. Currently these datasets can be accessed remotely through a federated call or locally by storing the datasets in a local triple-store.

SUMMR provides two sets of templates. One set is designed to perform the five mapping maintenance and reuse tasks. The other set is designed to perform the six mapping maintenance and reuse use cases. These two sets of templates are specialised for SCMR. When a user wishes to perform a mapping maintenance and reuse task or use case, the user would select the appropriate template and edit it to specify the
placeholder criteria in the template. The user would then execute the template via the SPARQL Processor, over the SCMRs and the datasets.

SUMMR assumes that the mappings it is performing maintenance and reuse over are already deployed by a dataset. For example, if a currently deployed mapping was found to be invalid (through a SUMMR template). If it is possible to repair the invalid mapping, it can be altered (changed) for the purpose of repair. The repaired mapping should now work correctly (no need to deploy a new mapping).

![Figure 5-1. Overview of the SUMMR Methodology.](image)

5.2. Elements of SUMMRs Design

This section presents the elements of the design of the SUMMR methodology. First the SPARQL Centric Mapping Representation (SCMR) is presented and examples of the representation are provided. Following that, the SUMMR templates are presented and examples of the templates, specialised to work with the SCMR, are also presented.

5.2.1. SPARQL Centric Mapping Representation (SCMR)

SUMMR provides a mapping representation named the SPARQL Centric Mapping Representation (SCMR) which is encoded in RDF. The SCMR is designed to represent two categories of mappings - vocabulary transformation category and interlink category of mappings. Representation of interlink category mappings is straightforward as an interlink typically consists of a single RDF triple. The vocabulary transformation category of mappings are more complex to represent as they specify how instance data is transformed from one vocabulary to another. The SCMR relies upon SPARQL CONSTRUCT queries to express mappings. SPARQL CONSTRUCT create new RDF graphs according to a graph template. As seen from the state of the art review (Chapter
3), CONSTRUCT queries have been proposed for use as vocabulary transformation mappings and in experimentation (Section 6.2), it has shown their effectiveness. To express interlink category mappings, the SCMR also relies upon SPARQL CONSTRUCT queries. In this way both categories of mapping use the same expression rather than a different one for each category. Using SPARQL CONSTRUCT queries provides a standards-based way to represent mappings, which should make users more comfortable with their use. In addition, SPARQL CONSTRUCT queries are executable on any standard SPARQL processor. This gives the SCMR the benefit of being executable and will make them useful outside of just mapping maintenance and reuse purposes. For example, in a data integration scenario, where data from multiple datasets is described according to multiple vocabularies need to be transformed into a single vocabulary.

SUMMR also uses standard SPARQL queries in order to perform maintenance and reuse over the actual mappings through its templates. As SCMR uses SPARQL CONSTRUCT queries to express a mapping, in order for the mappings to be queryable using SPARQL, the SPARQL CONSTRUCT query itself needs to be encoded in RDF. The SCMR uses the SPARQL Inferencing Notation (SPIN) [Kn11] SPARQL syntax for this purpose. SPIN, which is a W3C member submission, is an RDF rule language based on SPARQL queries. SPIN includes the SPIN SPARQL syntax, which is a vocabulary for encoding SPARQL queries in RDF. The SCMR includes the SPIN SPARQL syntax encoding of the SPARQL CONSTRUCT queries as part of the mapping representation.

Figure 5-2 provides an example of a SPARQL CONSTRUCT query, which is used to transform instance data of type ex:Class2 into type ex:Class1, encoded in RDF via the SPIN SPARQL syntax.

![Figure 5-2. Example of SPARQL CONSTRUCT Query Encoded in RDF via the SPIN SPARQL Syntax.](image-url)
As can be seen from the example, the SPIN SPARQL syntax encodes the SPARQL CONSTRUCT query at a **fine granularity**. The fine granularity means that no parts of the SPARQL CONSTRUCT query is represented as a string, all parts of the query are represented individually as RDF triples. This fine granularity affords greater analysis capabilities upon the SCMR by SPARQL queries, as will be seen in the experiment described in Section 6.3. However this fine granularity does come at the cost of an increase in the number of triples used to represent a mapping.

The SCMR provides three RDF vocabulary terms for modelling mappings. These terms are: the *Mapping* class, the *sparqlQuery* property and the *spinRepresentation* property. As a way of identifying an SCMR, they are modelled as an instance of the *Mapping* class. The *sparqlQuery* property is used to relate the SPARQL CONSTRUCT query as a string to the mapping (the string can be retrieved and executed). The *spinRepresentation* property is used to relate the RDF encoding (via the SPIN SPARQL syntax) of the SPARQL CONSTRUCT query to the mapping.

Previous research [Th13] has shown that meta-data annotated mappings can help with mapping reuse. The SCMR uses the Dublin Core Meta-data Initiative (DCMI) [We98] for meta-data fields. The DCMI provides a vocabulary of meta-data terms [Dc12] that are used to describe resources on the web. The DCMI vocabulary was chosen for meta-data terms as the vocabulary is well known and widely used. This is evidenced in a survey [Sc14] of the Linked Open Data Cloud which showed that the DCMI vocabulary was the fourth most widely used vocabulary in datasets in the Linked Open Data Cloud. The SCMR uses three meta-data terms from the DCMI vocabulary which are the *creator*, the *date* and *description*. The *creator* term is used to indicate who or what was responsible for the creation of a mapping. The *date* indicates when a mapping was created. *Description* provides a human readable description of the purpose of the mapping.

Below are two examples of the SCMR. The first example in *Figure 5-3* is an example of an SCMR representation of a vocabulary transformation category of mapping and the second example in *Figure 5-4* is an example of an SCMR representation of an interlink category of mapping. The purpose of the mapping in *Figure 5-3* is to transform instance data that is of class “film” from the Linked Movie Database datasets vocabulary, to the class ‘Film’ of the DBpedia dataset vocabulary. The mapping starts on line 05. The mapping identifier is called ex:vocabulary_mapping_1 and is an instance of the *scmr:Mapping* class. The mapping
meta-data is on lines 06-08. The SPARQL CONSTRUCT query is on line 09 and the RDF encoded CONSTRUCT query is related to the mapping identifier via the `scmr:spinRepresentation` property on line 10. Then the RDF encoded CONSTRUCT query is on lines 11-17.

```sparql
@prefix sp: <http://spinrdf.org/sp#>.
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix scmr: <https://www.scss.tcd.ie/~meehanal/scmr>.
@prefix ex: <http://www.example.org/example#>.

ex:vocabulary_mapping_1 a scmr:Mapping ;
dcterms:creator ex:person1 ;
dcterms:description "Used to transform data from LMDB film to DBpedia Film"^^xsd:string ;
dcterms:date "2016-08-17"^^xsd:date ;
scmr:sparqlQuery "CONSTRUCT{?s a <http://dbpedia.org/ontology/Film>}WHERE{?s a <http://data.linkedmdb.org/resource/movie/film>}" ;
scmr:spinRepresentation _:b1 .
_:b1 a sp:Construct;
sp:templates ([ sp:object dbpedia-voc:Film; sp:predicate rdf:type ;
sp:subject [ sp:varName "s"^^xsd:string ] ]);
sp:where ([ sp:object lmdb-voc:film ;
sp:predicate rdf:type ;
sp:subject [ sp:varName "s"^^xsd:string ] ]).
```

Figure 5.3. Example of a SCMR Vocabulary Transformation Category Mapping.

```sparql
@prefix sp: <http://spinrdf.org/sp#>.
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix scmr: <https://www.scss.tcd.ie/~meehanal/scmr>.
@prefix ex: <http://www.example.org/example#>.

ex:interlink_1 a scmr:Mapping ;
dcterms:creator ex:person1 ;
dcterms:description "Interlink Dr.Strabgelove from DBpedia to LMDB"^^xsd:string ;
dcterms:date "2016-08-17"^^xsd:date ;
scmr:sparqlQuery
"CONSTRUCT{<http://dbpedia.org/resource/Dr._Strangelove>
<http://www.w3.org/2002/07/owl#sameAs>
<http://data.linkedmdb.org/resource/movie/film/8>}WHERE{}}" ;
scmr:spinRepresentation _:b1 .
_:b1 a sp:Construct;
sp:templates ([ sp:object lmdb:8;
sp:predicate owl:sameAs;
sp:subject dbpedia:Dr._Strangelove; ]);
sp:where ().
```

Figure 5.4. Example of an SCMR Interlink Category Mapping.
The mapping in Figure 5-4 is concerned with interlinking the film Dr. Strangelove from the DBpedia dataset to the Linked Movie Database dataset. The mapping starts on line 05. The mapping identifier is called ex:interlink_1 and modelled as an instance of the scmr:Mapping class. The mapping meta-data is on lines 06-08. The SPARQL CONSTRUCT query is on line 09 and the RDF encoded CONSTRUCT query is related to the mapping identifier via the scmr:spinRepresentation property on line 10. Then the RDF encoded CONSTRUCT query is on lines 11-15.

In summary, SCMR can represent both vocabulary transformation and interlink categories of mappings. It uses the SPIN SPARQL syntax to encode in RDF, the SPARQL CONSTRUCT queries that are used to express the mappings. It also uses the Dublin Core Meta-data Initiative vocabulary for meta-data terms to annotate the mappings. Thus, the entire mapping represented using SCMR is queryable, for example by using SPARQL queries for mapping maintenance and reuse purposes. The additional benefit of the SCMR representation is that the mapping itself can be executed directly using a standard SPARQL processor.

5.2.2. SUMMR Templates

The motivation behind SUMMR providing query templates for mapping maintenance and reuse relates to the potential conferral of two advantages. First is that with such templates, mapping maintenance and reuse could be performed with greater ease and most likely quicker than having to create a new query to do so for the task. The second envisioned advantage is that templates provide a way for users to perform mapping maintenance and reuse who have little experience or lack the technical knowledge to create queries for such purposes.

Using standard SPARQL queries in templates offers three advantages. First it provides a standards-based way to performing mapping maintenance and reuse. Second, since they are standard SPARQL queries, users will likely be familiar with SPARQL and more comfortable with their use. Third is the templates are executable on a standard SPARQL processor, which eliminates the need for additional software to execute mapping maintenance and reuse tasks and use cases. The use of SPARQL templates has already been shown to be useful in different areas of research (not just the domain of mappings). For example, Kontokostas et al [Ko14] use SPARQL templates to represent data quality patterns that can be executed to discover inconsistencies in the quality of Linked Data Datasets.
The SUMMR methodology provides two sets of templates that rely on standard SPARQL queries, which can be specialised for different RDF-based mapping representations. The first set of templates have been designed for the mapping maintenance and reuse tasks that were derived from use cases identified from the state of the art. The second set of templates then are for the mapping maintenance and reuse use cases and their design influenced by the refined use cases described in Chapter 4. The rational for deriving the task templates is to show the flexibility of the approach. This flexibility allows the task to be combined together to perform use cases and an additional envisaged benefit of this flexibility is that the tasks can be combined together to perform new mapping maintenance and reuse use cases that may appear in the future.

5.2.2.1. Maintenance and Reuse Task Templates

For the first set of templates, SUMMR provides five templates designed for the five mapping maintenance and reuse tasks previously derived (from the state of the art review) at a one-to-one relationship. These five task templates rely on standard SPARQL queries and are designed to be performed over an RDF-based mapping representation. Table 5-1 shows the SUMMR task templates and their purpose. While there are five task templates, each one is flexible with the exception of Task Template 3 (as seen in the examples below). They can vary based on factors such as whether vocabulary transformation or interlink category mappings are to be considered at the same time or each separately, how general or specific the mapping search criteria has to be or whether a federated SPARQL call is needed to access a remote dataset. This flexibility allows the templates to be useful in a range of scenarios where the needs of users differ. Note also that task templates can be combined to form complex operations and also are used in combination to implement the maintenance and reuse use cases.

<table>
<thead>
<tr>
<th>SUMMR Template ID</th>
<th>Task Name / Template Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Template 1</td>
<td>Discover a mapping based on search criteria</td>
</tr>
<tr>
<td>Task Template 2</td>
<td>Discover and retrieve individual properties of a mapping</td>
</tr>
<tr>
<td>Task Template 3</td>
<td>Retrieve the entirety of a mapping</td>
</tr>
<tr>
<td>Task Template 4</td>
<td>Compare mapping properties to resources from a dataset</td>
</tr>
<tr>
<td>Task Template 5</td>
<td>Alteration of a mapping</td>
</tr>
</tbody>
</table>

Table 5-1. SUMMR Mapping Maintenance and Reuse Task Templates
An example of each of the SUMMR task templates are provided below specialised for SCMR. Variables within the queries that begin with the ‘$’ (standard syntax for declaring variables in SPARQL) character represent placeholders in the queries. This means that these variables may need to be replaced by a user with specific criteria. Variables that begin with the ‘?’ (standard syntax for declaring variables in SPARQL) character should not be replaced.

**SUMMR Task Template 1:**
The example in Figure 5-5 is an example of SUMMR Task Template 1 specialised for SCMR. This template is concerned with discovering vocabulary transformation category mappings with specific search criteria. Depending on what criteria is to be searched for and how general or specific this criteria is to be, the template can vary.

For example, line 05 contains a triple-pattern where meta-data search criteria is specified in this template. This line is optional (depending if meta-data is to be used as search criteria for a mapping) or can be repeated for the searching of multiple meta-data fields. In this example, the variable $METADATA_PREDICATE will be a URI (URI of a Dublin Core meta-data term) and the variable $METADATA_OBJECT will be a literal.

The next search criteria is related to the source and target of the mapping. Depending on how specific or general this search criteria is to be, the query will vary. An example of specific search criteria would be finding mappings between specific vocabulary terms, such as the film class from the DBpedia dataset as the source of the mapping and the film class from the Linked Movie Database dataset as the target. An example of general search criteria would be finding mapping that have a DBpedia vocabulary term as the source and a Linked Movie Database vocabulary term as the target. Lines 09 and 10 are where source search criteria is specified. Lines 11 and 12 are where target search criteria is specified. Each of these two sets of lines are optional, so if none were included, mappings would be searched for based on meta-data only. Also, if only one set is used, for example the source search criteria, then mappings will be searched for based on source search criteria and the target of the mapping is not considered.
PREFIX sp: <http://spinrdf.org/sp#>
PREFIX scmr: <https://www.scss.tcd.ie/~meehanal/scmr>
SELECT ?mapping
?mapping $METADATA_PREDICATE $METADATA_OBJECT.
?source ?p $SOURCE_SEARCH_CRITERIA.
FILTER ( REGEX( STR($SOURCE_SEARCH_CRITERIA) , $SOURCE_DATASET_DOMAIN ) )
?target ?p1 $TARGET_SEARCH_CRITERIA.
FILTER ( REGEX( STR($TARGET_SEARCH_CRITERIA) , $TARGET_DATASET_DOMAIN ) )
?trans ?p2 $TRANSFORMATION_SEARCH_CRITERIA.}

Figure 5-5. SUMMR Task Template 1 Specialised for SCMR.

If a mapping was to be searched for based on specific source search criteria, the variable $SOURCE_SEARCH_CRITERIA on line 09 would be set as a URI of the specific vocabulary term to be searched for and line 10 would be removed. If a mapping was to be searched for based on a general source search criteria, the variable $SOURCE_SEARCH_CRITERIA would remain untouched and the $SOURCE_DATASET_DOMAIN variable would be a string containing the domain URI of the source dataset. The same applies for searching for mappings with specific or general target search criteria. In addition, it is also possible to have either the source or target of the mapping be searched based on specific or general search criteria. Line 13 allows a user to search for mappings that contain a specific data transformation. This line is optional (depending if a mapping is to be searched for on transformation criteria). It is done by specifying the $TRANSFORMATION_SEARCH_CRITERIA variable with a URI of the transformation (concatenation, multiple, divide) described by the SPIN SPARQL Syntax (for example sp:concat, sp:mul, sp:divide).

SUMMR Task Template 2:
The example in Figure 5-6 is an example of SUMMR Task Template 2 specialised for SCMR. This template is concerned with discovering and retrieving the meta-data, the source and the target properties of a specific vocabulary transformation category mapping. The only criteria in this template to specify is the $MAP_ID variable. The $MAP_ID variable will be a URI (URI which will be a mapping identifier).
SUMMR Task Template 2 Specialised for SCMR

SUMMR Task Template 3:

Figure 5-7 displays an example of SUMMR Task Template 3. This template simply returns all data related to a mapping. In this template, the $MAP_ID variable will be a URI.

SUMMR Task Template 4:

The example in Figure 5-8 is an example of SUMMR Task Template 4 specialised for SCMR. This template is concerned with comparing the source and target properties of a mapping to the properties in a source and target dataset to see if the properties exist in those datasets.

In this example, the target properties, represented by the variable $MAP_TARGET_PROPERTIES, are compared to the target dataset that is accessed through a federated call. The $MAP_TARGET_PROPERTIES variable will be a URI and the $REMOTE_SPARQL_ENDPOINT variable will also be a URI of the remote SPARQL endpoint.

The source properties, represented by the variable $MAP_SOURCE_PROPERTIES, are compared to the source dataset that is stored in a named graph in a triple-store. The $MAP_SOURCE_PROPERTIES variable will be a URI and the $NAMED_GRAPH variable will also be a URI of the named graph.

It is possible to compare both source and target properties to datasets through federated calls or datasets stored in named graphs. It is also possible to only compare source properties or target properties from mappings to a dataset.
SUMMR Task Template 5:

Figure 5-9 displays an example of SUMMR Task Template 5 specialised for SCMR. This template is concerned with changing a property in a vocabulary transformation category mapping. In this template, the criteria to specify is the mapping identifier specified as the variable $MAP_ID. The remaining criteria to specify is the property to delete, specified as variable $MAPPING_PROPERTY_TO_DELETE and the property to insert, specified by the variable $MAPPING_PROPERTY_TO_INSERT. All variables in this template will be URIs.

The task templates are not always useful by themselves. For example, Task Template 4 will just compare a mapping property, that a user must specify, to a dataset and will return true or false based on whether the property exists in the dataset or not. However, the combination of Task Templates 1, 2 and 4 will allow mappings with specific search criteria to be checked to see if they are invalid in relation to the datasets that the mapping references. The task templates can be combined to form more complex operations, and are used in combination to implement maintenance and reuse use cases.
5.2.2.2. Maintenance and Reuse Use Case Templates

For the second set of templates, SUMMR provides six templates for the six mapping maintenance and reuse use cases with again a one-to-one relationship. These six templates are made from combinations of the task templates describe in the previous sub-section, and their design influenced by the six refined use cases from Chapter 4.

Table 5-2 lists the SUMMR Use Case Templates, the use case the template performs, the template category and which task templates combine to make the use case template. The use case templates were separated into two categories. Four of the templates fall into the reuse category and two into the maintenance category. As can be seen, the use case templates are made up of two or more task templates with the exception of use case template 6. Use Case Template 6 is for the repair of an invalid mapping, which requires the deletion of invalid properties from the mapping and replacing them with new valid properties. Therefore Use Case Template 6 only uses Task Template 5 which is for the alteration of a mapping. Similar to task templates, the use case templates are designed to be flexible which allow them to be meet the needs of different users.

<table>
<thead>
<tr>
<th>SUMMR Template ID</th>
<th>Use Case Name / Template Purpose</th>
<th>Category</th>
<th>Task Template Involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Template 1</td>
<td>Retrieve specific mappings for sharing</td>
<td>Reuse</td>
<td>1, 3</td>
</tr>
<tr>
<td>Use Case Template 2</td>
<td>Discover Information from Existing Mappings to Aid in the Creation of New Mappings</td>
<td>Reuse</td>
<td>1, 2</td>
</tr>
<tr>
<td>Use Case Template 3</td>
<td>Discover mapping paths to aid in the creation of new mappings</td>
<td>Reuse</td>
<td>1, 2</td>
</tr>
<tr>
<td>Use Case Template 4</td>
<td>Discover back links to aid in the creation of new mappings</td>
<td>Reuse</td>
<td>1, 2</td>
</tr>
<tr>
<td>Use Case Template 5</td>
<td>Discover invalid mappings due to changes in datasets a mapping references</td>
<td>Maintenance</td>
<td>1, 2, 4</td>
</tr>
<tr>
<td>Use Case Template 6</td>
<td>Repair of an invalid mapping</td>
<td>Maintenance</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 5-10 below provides another depiction of the information displayed in Table 5-2, where it displays which of the SUMMR Task Templates are combined together to make up the SUMMR Use Case Templates. From this table and figure, it should be clear to the reader that the approach designed is very flexible, allowing for potentially
more task templates to be added at a future point and allow for more use case templates to be defined based on the task templates.

![Diagram of SUMMR Task Templates and Use Case Templates](image)

**Figure 5-10. Depiction of SUMMR Task Templates that combine to create SUMMR Use Case Templates.** The different coloured arrows emanating from the Task Templates point towards the Use Case Templates that they combine together to make up.

In the following text, examples are presented of SUMMR use case templates, Use Case Template 1 and 5, representative examples from each of the two categories of reuse and maintenance. See the *Appendix A* for examples of the remaining use case templates.

**SUMMR Use Case Template 1:**

*Figure 5-11* displays an example of SUMMR Use Case Template 1 specialised for SCMR. This template is concerned returning entire SCMR vocabulary transformation category mappings that adhere to specific mapping source and target search criteria. The search criteria explained in SUMMR Task Template 1 can be applied to this template, but is left out in this example. The variables `$SOURCE_SEARCH_CRITERIA` and `$TARGET_SEARCH_CRITERIA` would be set as the URIs of the specific vocabulary terms to be searched for.
SUMMR Use Case Template 5:

Figure 5-12 displays an example of SUMMR Use Case Template 5 specialised for SCMR. This template is concerned with discovering invalid SCMR interlink category mappings. This particular template includes a federated SPARQL call to a dataset (related to the target of the interlink) and assumes that the other dataset (related to the source of the interlink) is stored in a named graph in a local triple-store. The criteria that needs to be specified in this template are specified in the following four variables. The $TARGET_DATASET_DOMAIN and the $SOURCE_DATASET_DOMAIN will both be strings, indicating the domain URI of the target and source datasets respectively. The $REMOTE_SPARQL_ENDPOINT and the $NAMED_GRAPH variables will both be URIs indicating the URI of the remote SPARQL endpoint where the target dataset is located and the URI of the named graph where the source dataset is located.
5.2.3. SUMMR Templates Summary

In summary, the SUMMR methodology provides two sets of templates that use standard SPARQL queries. One set is of five templates designed for the five mapping maintenance and reuse tasks derived from the use cases. The other set is of six templates for the six mapping maintenance and reuse use cases identified from the state of the art and are their design influenced by the six refined use cases from Chapter 4. The intention with the templates is to remove much of the need for new queries to be created when performing mapping maintenance and reuse, with the likely result of allowing such tasks and use cases to be performed quicker. The templates should also support users with little experience to perform maintenance and reuse of mappings.

5.3. Operation of the SUMMR Methodology

This section describes how the SUMMR methodology would be utilised for mapping maintenance and reuse. It is envisaged that SUMMR will be of benefit to Linked Data dataset providers and maintainers in two situations. First, it will be of use for those looking to discover new mappings related to datasets of interest - through the reuse of existing ones. Second it will be of use for those who wish to discover and repair existing mappings that have become invalid due to changes within the dataset they maintain, or external datasets the mappings reference. It is assumed in this section that the SCMR is used to represent mappings and the SUMMR templates have been specialised for SCMR.

The operation of SUMMR can be undertaken in two ways. One way is through manual operation, where SCMRs will be manually created and SUMMR templates will be manually edited and executed. The other approach of operation would be through a software tool set, which implement aspects of SUMMR. While this thesis is not directly concerned with the design and implementation concerns of an entire tool set, it will be briefly discussed as a possible SUMMR implementation in the following section.

The following provides an overview of how a user, who wishes to perform mapping maintenance and reuse would utilise SUMMR. The operation is depicted as a flow diagram in Figure 5-13 and described below:

1. First, the SCMR would be adopted. This would mean any new mappings created would be represented in SCMR and any existing mappings should also be converted to SCMR. The SCMRs would then be stored in a triple-store with a SPARQL processor which the user has access to. In reality, manual conversion
of a large number of mappings to SCMR would be unreasonable, therefore a tool to aid with this conversion would be necessary.

2. Next step would involve a user deciding on which mapping maintenance or reuse use case that is to be performed. Based on this decision, the user would then select the SUMMR template that corresponds to that use case. SUMMR Use Case Templates 1-5 would be stored as “.rq” (file extension for SPARQL queries) files and SUMMR Use Case Template 6 would be stored as a “.ru” (file extension for SPARQL UPDATE queries) file. With no tool support, the user would manually select and make a copy of the template file.

3. This is a conditional step based on if a user has chosen the use case to “Discover invalid mappings due to changes in datasets a mapping references”. If the user wishes to compare mappings to a dataset that will be accessed from the same triple-store that the mappings are in (and the dataset is currently not stored there) then the user must load that dataset into the triple-store.

4. For the next step, the template (within the query file) would then be edited, through the use of a simple text editor, with the criteria required for the template.
5. Finally, the user would execute the query file via the SPARQL processor.

5.3.1. Operation of SUMMR in a Mapping Lifecycle

This sub-section presents the wider context of SUMMR and shows where it fits and operates in the overall lifecycle of mappings. The OISIN (Ontology Interoperability in Support of Semantic Interoperability) framework [Su07], is a process framework with the overall aim to reduce the time and effort involved in creating mappings between ontologies. Although originally designed for mappings between ontologies, the lifecycle is equally applicable to mappings between Linked Data datasets.

Part of the OISIN framework is a mapping lifecycle which breaks down the mapping process in different phases and it has been further refined and extended by Thomas in 2013 [Th13]. This extended OISIN mapping lifecycle provides a more comprehensive breakdown of the phases and actions involved in the mapping process, compared to other proposed mapping lifecycles [Sh13], making it a reasonable context in which to show where SUMMR fit in overall. The extended OISIN mapping lifecycle is displayed in Figure 5-14.

![Extended OISIN Mapping Lifecycle diagram (from Thomas [Th13]).](image)
The extended OISIN mapping lifecycle is divided into four phases – the characterisation phase, the matching phase, the mapping phase and the management phase.

The phases of particular note are the mapping phase and the management phase. The mapping phase is concerned with creating actual mappings from the alignments created in the matching phase. It is in this phase that SCMRs would be created. The management phase is concerned with the management of the stored mappings created in the previous phase. Here decisions are made on finding mappings for reuse and sharing, and the alteration of mappings if they need to be updated.

Figure 5-15 displays where the SUMMR methodology operates in a refined extended OISIN mapping lifecycle, which explicitly shows the role of maintenance and reuse in the lifecycle. SUMMR operates in the management phase of the lifecycle, where it is concerned with reuse and maintenance of the existing mappings that are stored. SUMMRs operation is encompassed within the grey area of the management phase and subsumes the operations of the management phases from the previous lifecycle. The mapping reuse area in the figure is related to performing the mapping reuse use cases. The orange arrows in the figure are related to mapping reuse steps and decisions. The mapping maintenance area in the figure are related to performing mapping maintenance use cases. The green arrows in the figure are related to mapping maintenance steps and decisions.

Figure 5-15. Extended OISIN mapping lifecycle with the SUMMR Methodology.
In summary, this sub-section presented where the SUMMR methodology operates in the overall lifecycle of mappings. By doing so, a further extension of the OISIN lifecycle has been created which explicitly shows the role of maintenance and reuse in the overall lifecycle.

5.4. SUMMR Implementation

This section presents a partial implementation of SUMMR that has been developed named the SUMMR Interlink Validation Tool and discusses other possible implementations that could be developed for SUMMR.

5.4.1. SUMMR Interlink Validation Tool

The SUMMR Interlink Validation Tool is a prototype, command line, java-based, open-source tool\(^9\). It implements SUMMR Use Case Template 5 (which includes Task Templates 1, 2 and 4) which is the use case designed to discover invalid mappings due to changes in datasets that the mappings reference. It was developed to test (part of) SUMMRs usefulness in a real scenario - that is for discovering invalid interlink category mappings between Linked Data datasets.

Requirements for the tool were drawn from the interlink management process of the DBpedia dataset but can be used to validate interlink category mappings between any Linked Data datasets. The DBpedia interlink management process is described further in the case study in Section 6.5. The Interlink Validation Tool is designed to perform mapping maintenance and reuse use case 5 in the particular situation where a specific source dataset is being maintained. This source dataset contains interlink category mappings to external target datasets. As the source dataset and the target datasets evolve over time, the maintainers of the source dataset need to ensure, that none of the existing interlink category mappings have become invalid due to the evolution of the datasets.

The tool consists of three components. Two of the components are internal: the Template Processor and the Data Loader and Remover. The remaining component is an external component and is the Parameter File. The tool relies on a triple-store and a SPARQL processor. The Parameter File is where the details are specified about the interlink category mappings that are to be validated between the Source Dataset and

\(^9\) https://github.com/aligned-h2020/ALIGNED_Code/tree/master/interlink_validation_tool
remote external dataset. The operation and architecture of the tool is presented in Figure 5-16 below.

![Diagram of SUMMR Interlink Validation Tool]

**Figure 5-16. Operation of the SUMMR Interlink Validation Tool. The arrows indicate the flow of information/data among the different components.**

The tool contains two specialisations of SUMMR Use Case Template 5 - specialised for interlink category mappings represented as RDF triples or in SCMR. The *Data Loader and Remover component* is responsible for loading and removing interlink category mappings and external datasets into the triple-store. The *Template Processor* is responsible for creating a specialisation of SUMMR Use Case Template 5 - based on the parameters specified in the *Parameter File* and how the interlink category mappings are represented. For interlink category mappings represented as RDF triples and in SCMR, the appropriate specialisation of SUMMR Use Case Template 5 is used. The *Template Processor* is also responsible for sending SUMMR Use Case Template 5 to the SPARQL processor for execution and then writing the results to an *Invalid Interlinks* file.

The operation and set up of the tool is described below. First the parameters in the *parameter file* need to be specified. These parameters are:

1. The name of the external dataset.
2. The location of the interlink category mappings in the *Local Triple-store* (a URI of the named graph the interlink category mappings are in) or the file path of a set of interlink category mappings if they are stored in an external file.
3. An indicator specifying whether or not the external dataset instances of the interlink category mappings are to be validated. If they are not to be validated,
then only the local, source instances will be validated. If they are to be validated, then a user specifies how the instances will be validated against the respective external dataset. An external dataset can be accessed through a federated SPARQL query or a Dataset Dump file.

4. Therefore, the fourth parameter can be the URI of an external SPARQL endpoint or a file path to the Dataset Dump file.

When all parameters are set, the tool can be run. The following explains the procedure undertaken by the tool when it is run:

- The tool first checks where the interlink category mappings are to be validated, if the interlink category mappings are in an external file, the tool will load them into a named graph in the triple-store.
- Next, based on the third parameter in the Parameter File, a specialisation of SUMMR Use Case Template 5 is generated - for example, if the external dataset is to be accessed via a federated query, then a federated query call will be included in the query template, with the external SPARQL endpoint URI from parameter 4 being used. If the third parameter specifies that an external dataset is to be accessed from a Dataset Dump file, then this dump file is loaded into a named graph in the triple store.
- Next the Template Processor sends the query template to the SPARQL processor of the Local Triple-store for execution. The source instances are always checked against the Source Dataset. The execution results are then returned to the Template Processor which outputs the results to the Invalid Interlinks file and the tool removes any data it temporarily loaded into the Local Triple-store. The tool will repeat for each set of interlink category mappings that are to be validated specified in the Parameter File.

In summary, the SUMMR Interlink Validation Tool implements SUMMR Use Case Template 5 for discovering invalid interlink category mappings between Linked Data datasets. The tool was used in the case study of this thesis to validate the interlink category mappings that were published in the v.2015-10 DBpedia dataset release. It is now part of multiple technologies used by the DBpedia project during the release process of a new version of the dataset. The tool will be useful for the maintainers of Linked Data datasets, who wish to detect if any of the interlink category mappings that they maintain are invalid. In its current design (a command line tool) it is inclined
towards technical users. However, with the design of a graphical user interface, it could potentially be of benefit for less technically inclined users.

5.4.2. Potential SUMMR Tools

This sub-section discusses potential tools that could support the use of the SUMMR methodology. Potential tools will vary based on the level of support they provide and the type of user catered for (technical or non-technical). Tools for non-technical users would likely require a more sophisticated user interface design, whereas for technical users it could remain simple.

A possible tool would be for the creation of SCMRs. A SCMR creation tool would allow users to provide an identifier for a mapping and input the mapping meta-data. A user would also input a SPARQL CONSTRUCT query and the tool would convert this query to RDF via the SPIN SPARQL syntax. The tool would then output a SCMR like the examples provided in Section 5.2.1. Additional support could be incorporated into this tool to generate SPARQL CONSTRUCT queries for users with limited knowledge of mapping creation.

Other possible tools would be for the selection, editing and execution of SUMMR templates. Tools could be created to support a particular use case (like the SUMMR Interlink Validation Tool) or multiple use cases. They could be command line based (again like the SUMMR Interlink Validation Tool) or provide a graphical user interface. Tools with a graphical user interface could allow for specific templates to be chosen and indicate the criteria to be specified for a template and allow that criteria to be entered. The templates could then be executed with the press of a button. More sophisticated tools could incorporate error handling when editing templates and indications which specify the type of criteria allowed for placeholders in specific templates.

These possible tools discussed above could be combined together into an entire tool set that fully supports SUMMR’s functionality. This tool set could also allow for incorporation of ontology matching tools, which would be useful in the process of repairing invalid mappings.

5.4.3. SUMMR Implementation Summary

In summary, this section has presented the SUMMR Interlink Validation Tool which is an implementation of SUMMR for discovering invalid interlink category mappings between Linked Data datasets. While this thesis is not concerned with the design and
implementation of a comprehensive tool set for the SUMMR methodology, aspects of such a possible tool set was also discussed.

5.5. Reflections upon the Design of SUMMR

This section reflects upon the design of the SUMMR methodology, examining its features and limitations (Section 5.5.1) and its potential specialisation to other domains (Section 5.5.2).

5.5.1. Features and Limitations

The design of the SUMMR methodology uses standard SPARQL queries for two parts of its design. The SCMR relies upon standard CONSTRUCT queries to represent vocabulary transformation and interlink category mappings. The templates that SUMMR provides rely upon standard ASK, DESCRIBE, SELECT and UPDATE queries. The use of CONSTRUCT queries allows for SCMRs to be executed on a standard SPARQL processor. This makes the SCMR useful beyond maintenance and reuse purposes. The use of standard ASK, DESCRIBE, SELECT and UPDATE queries for SUMMR’s maintenance and reuse templates also means that the (mapping maintenance and reuse) tasks and use cases can be executed on a standard SPARQL processor. In addition, as seen from the state of the art review of existing approaches that support mapping maintenance, reuse or both, none are able to perform all six of the mapping maintenance and reuse use cases. The SUMMR methodology does support all six of the use cases as evidenced by the evaluations in Chapter 6.

The SUMMR methodology is likely to be of most benefit to the providers and maintainers of Linked Data datasets. These users are likely to have a triple-store and SPARQL processor in deployment. SUMMR specifies that mappings and the datasets that the mappings reference are to be stored in and accessed from a triple-store with a SPARQL processor. Thus the design of SUMMR reduces the need for additional software to store and manage mappings and datasets and perform reuse and maintenance over the mappings and datasets. It is hoped that SUMMRs emphasis on the reuse of standard Linked Data infrastructure for mapping maintenance and reuse, and provision of reusable SPARQL templates will make users more comfortable with the methodology and make it more appealing for adoption and use.

A potential step involved in the repair of invalid mappings is to discover new alignments between resources in updated datasets and use the information in these alignments to repair the mapping. The SUMMR methodology itself is not concerned
with discovering alignments. However, SUMMR would be suitable to be used in conjunction with ontology matching tools. These tools would be used to discover new alignments and the information from these alignments could be input to SUMMR Use Case Template 6 for repairing an invalid mapping.

Existing approaches to discovering mappings that have become invalid in relation to the datasets that they reference employ techniques such as constantly monitoring datasets for changes to resources or performing a comprehensive difference check of resources between two versions of a dataset. The results of these techniques are then used to detect mappings that contain changed resources from the datasets. SPARQL does not support such comprehensive changes to datasets. Due to this, the SUMMR approach to discovering invalid mappings (entailed in SUMMR Use Case Template 5) does have a limitation. SUMMR Use Case Template 5 operates by directly comparing mapping properties to dataset instances and vocabulary terms from a dataset. Two problems that can occur when a dataset evolves are the resource merge and resource split problem [Do12]. A resource merge can involve two or more resources merging into a single resource and a resource split can involve an existing resource splitting into two or more resources. A situation can arise where a resource merge or split takes place and the original resource identifier does not change, so a resource may have changed semantically, but syntactically (through an unchanged resource identifier) it is the same. It is that situation where a resource changes semantically but not syntactically is where SUMMR approach can fail to correctly detect an invalid mapping.

5.5.2. Specialising SUMMR for Mapping Maintenance and Reuse for Other Domains

The SUMMR methodology is designed for Linked Data mapping maintenance and reuse and specifies technologies such as SPARQL. This sub-section briefly discusses a possible abstraction of SUMMR, outlining what a query-based methodology for mapping maintenance and reuse might look like. This higher level abstraction could possibly be used to develop other specialisations of the methodology, not just using a triple-store and SPARQL or not just for Linked Data domain.

It is envisaged that such a query-based methodology should select a mapping representation to begin with. This representation would have three characteristics as depicted in Figure 5-17. The first characteristic is that the mapping should be represented as a query from an open standard query language. The second characteristic
would be that the mapping representation itself should be queryable using a standard query language. The third characteristic of the mapping representation is that is must support annotation.

A query-based methodology would propose the use of templates that would perform maintenance and reuse over the mappings and the datasets that the mapping references. These templates should be created using an open standard query language, preferably the same query language used to represent the mappings.

![Figure 5-17. Characteristics of the Proposed Mapping Representation to be provided by the Query-based Methodology.](image)

These templates should be designed to perform the useful mapping maintenance and reuse tasks and use cases (with exception to Use Case 4: Discover Back Links to Aid in the Creation of New Mappings, as use case 4 is likely only applicable to the Linked Data domain) as outlined in Chapter 3. The templates should contain placeholders for information that will need to be provided by the user of the methodology.

In summary, an abstracted query-based methodology (see Figure 5-18) would advocate:

- Adopting a mapping representation where the mapping is expressed as (i) a query from a standard query language, (ii) having that query be queryable and (iii) that the representation would support annotation in order to capture meta-data related to the mapping.
- The provision of query templates that support mapping maintenance and reuse tasks and use cases.
5.6. Chapter Summary
This chapter has introduced the SUMMR methodology for performing maintenance and reuse for mappings between Linked Data datasets. The breakdown of SUMMR into a set of SPARQL query templates for mapping maintenance and reuse and the SCMR (SPARQL Centric Mapping Representation) was presented. The operation of SUMMR was discussed and the OISIN mapping lifecycle framework was refined to show where the operation of SUMMR fits into an overall mapping life cycle. SUMMR has been prototyped as the open-source SUMMR Interlink Validation Tool used in the DBpedia case study (Chapter 6). The chapter finished with reflections upon the design of SUMMR, examining its features which also proposes how SUMMR could be abstracted to a high level design for a query-based methodology for mapping maintenance and reuse.
6. Evaluation

This chapter contains descriptions of four evaluations of the SUMMR methodology. These four evaluations consist of three experiments and one case study. Section 6.1 provides an introduction and overview of each of the evaluations in the chapter. Section 6.2 presents the first experiment which involved evaluating the SCMR expressivity for representing vocabulary transformation category mappings and encoding them in RDF. Section 6.3 presents experiment 2 which evaluated the ability of the SUMMR methodology templates to perform tasks one to three of the five mapping maintenance and reuse tasks derived in Chapter 3. Section 6.4 presents the third experiment which evaluates the SUMMR methodology templates to perform mapping maintenance and reuse tasks four and five through discovering and repairing invalid mappings. In Section 6.5 a SUMMR case study is presented. This was designed to show the usefulness of the SUMMR methodology by applying it in a real world Linked Data dataset situation. Specifically, the case study describes the application of the SUMMR Interlink Validation Tool to the interlink management process of the DBpedia dataset. The chapter summary is presented in Section 6.6.

6.1. Introduction

This chapter contains descriptions of the three experiments and one case study that were undertaken to evaluate the proposed SUMMR methodology. Provided in this section is an overview, pointing out the purpose and what was involved in each.

A point to note regarding the datasets used in the evaluations of this thesis. The datasets are used for the purposes of testing the expressivity of the SCMR with regards to representing vocabulary transformation category mappings and testing SUMMR templates to perform mapping maintenance and reuse tasks over mappings. For these purposes, no benchmarks or benchmark datasets exist, therefore no benchmarks are used for comparison in the evaluations. A notable benchmarking event within the domain of matching and mapping is the Ontology Alignment Evaluation Initiative (OAEI) [Oa17]. The OAEI is run annually to test the capabilities of ontology matching algorithms through checking if these algorithms can correctly discover gold standard alignments (which are manually created by experts) between source and target ontologies. Multiple ontologies are used each year which pose different challenges, for example ontologies which cover different domain and ontologies in different natural languages. Ontologies and alignments between ontologies are made available from
previous OAEI events as benchmarking datasets that people can use to test their ontology matching algorithms. Since these alignments are not converted into mappings, they are not suitable for use in the evaluations in this thesis.

**Experiment 1:**
The purpose of the first experiment is to evaluate two aspects of the mapping representation that SUMMR provides - the SPARQL Centric Mapping Representation (SCMR). The first aspect evaluated is the expressivity of the SCMR in regards to representing executable vocabulary transformation category mappings. This was done by comparing the expressivity of SPARQL CONSTRUCT queries (which the SCMR utilises to express vocabulary transformation category mappings) against the R2R Mapping Language. As stated in Section 3.5 of the state of the art review, the R2R Mapping Language is one of the mapping representations that stand out due to its characteristics. To recall these characteristics, they are: is encoded in RDF, provides its own vocabulary to model and describe the mapping, provides mapping meta-data and is executable with regards to executing vocabulary transformation category mappings. The SCMR shares these characteristics and similarities between the two can be seen in the examples of the SCMR in Figure 5-3 and the R2R Mapping Language in Figure 3-5. Considering these similarities and that the R2R Mapping Language is specifically designed to represent vocabulary transformation category mappings, the R2R Mapping Language is the best candidate, of the existing approaches from the state of the art, to compare the expressivity of the SCMR in regards to representing executable vocabulary transformation category mappings.

The second aspect evaluated is the expressivity of the SCMR in regards to encoding vocabulary transformation category mapping in RDF. This was done by testing the expressivity of the SPIN SPARQL syntax (which the SCMR utilises) when encoding SPARQL CONSTRUCT queries in RDF.

**Experiment 2:**
The purpose of the second experiment is to test the mapping maintenance and reuse affordance of the SCMR and the ability of the SUMMR methodology query templates to support a potential user to perform mapping maintenance and reuse task numbers 1 to 3. Table 6-1 presents the five mapping maintenance tasks. Tasks 1 to 3 are evaluated in this experiment.
This evaluation was done by first deriving a set of 12 mapping retrieval use case (which encompass mapping maintenance and reuse task numbers 1 to 3), originally motivated in the research of Thomas et al. [Th11]. Two sets of bespoke SPARQL queries were designed to perform the twelve mapping retrieval use cases over mappings represented in SCMR and in the R2R Mapping Language. Then SUMMR Task Templates 1 to 3 were used in combination to perform each of the twelve mapping retrieval use cases. The SUMMR templates were specialised for both SCMR and the R2R Mapping Language.

The bespoke queries and the two specialisations of the SUMMR templates were all executed and their execution results analysed to determine the affordance of the SCMR and the ability of SUMMR templates to perform mapping maintenance and reuse task numbers 1 to 3.

The R2R mapping language was used again in this experiment as due to its similarities with the SCMR as stated above (under Experiment 1), it is a good candidate to create specialisations of SUMMR templates for. A difference between the two representations is examined in this experiment, which is the affordance offered by a fine grain representation (SCMR) versus a more coarse grain representation (R2R Mapping Language) for performing mapping reuse tasks using SUMMR templates.

**Experiment 3:**
The purpose of the third experiment is to test the ability of the SUMMR methodology query templates to support a potential user to perform mapping maintenance and reuse task numbers 4 and 5. This was done by using SUMMR Use Case Template 5 to discover invalid vocabulary transformation and interlink category mappings both represented in SCMR. The purpose of SUMMR Use Case Template 5 is to “Discover invalid mappings due to changes in datasets a mapping references” and it involves
mapping maintenance and reuse tasks 1, 2 and 4. The invalid mappings discovered by Use Case Template 5 are compared to a gold standard to determine the templates effectiveness.

SUMMR Use Case Template 6 was then used to repair the invalid mappings discovered (from SUMMR Use Case Template 5). The purpose of SUMMR Use Case Template 6 is for the “Repair of an invalid mapping” and it involves mapping maintenance and reuse task 5. The execution results of the repaired mappings were compared to a gold standard to determine the effectiveness of SUMMR Use Case Template 6.

Case Study:
The purpose of the case study was to show the usefulness of the SUMMR methodology in a real Linked Data dataset management scenario. This was done by applying SUMMR Use Case Template 5, via the SUMMR Interlink Validation Tool, to the DBpedia dataset release process. The SUMMR Interlink validation Tool was used to validate the interlink category mappings in the DBpedia v.2015-10 release.

6.2. Experiment 1: Evaluation of the SPARQL Centric Mapping Representation
This section presents the first evaluation of the SUMMR methodology. Section 6.2.1 presents an introduction to and motivation for the experiment. Section 6.2.2 presents the experiment hypotheses. A description of the datasets that were used in this experiment is presented in Section 6.2.3. The methodology undertaken for the experiment is described in Section 6.2.4. Section 6.2.5 presents the procedure taken. Section 6.2.6 presents the results obtained and a discussion of those results are given in Section 6.2.7. The section finishes with a conclusion in Section 6.2.8

6.2.1. Introduction
Mapping maintenance and reuse activities are performed over mappings. Therefore it is important to use a mapping representation that can represent the mappings under consideration for maintenance and reuse. A mapping representation that cannot adequately represent mappings could be missing properties of a mapping. Such properties include, the source or target of a mapping, any data transformations involved and meta-data terms. These missing properties may need to be accessed for a mapping maintenance or reuse task or use case. For example, consider a mapping representation
that cannot represent data transformations that involves the concatenation of text strings. It would not be possible to discover mappings that contain concatenation data transformations. Therefore, to effectively perform mapping maintenance and reuse, it is important to use a mapping representation that can represent the mappings under consideration.

This lab-based experiment tests the expressivity of the SPARQL Centric Mapping Representation (SCMR) in two parts. The first part is concerned with the expression of vocabulary transformation category mappings in SPARQL and the second part is concerned with the encoding of SPARQL-based vocabulary transformation category mapping in RDF. This involves testing the ability of SPARQL CONSTRUCT queries to represent vocabulary transformation category mapping and the SPIN SPARQL syntax to encode those SPARQL CONSTRUCT queries in RDF, two technologies that the SCMR uses. While the SCMR also represents interlink category mappings, this experiment is not concerned with evaluating its expressivity for this category. Representing interlinks is straight forward as an interlink typically consists of a single RDF triple. Vocabulary transformation category mappings on the other hand are more complex, as they are executable mappings that are used to transform data between the vocabularies of Linked Data datasets. Vocabulary transformation category mappings consist of two or more RDF triples (at least one triple each related to the source and target of the mapping) which specify how instance data is transformed from the vocabulary of one Linked Data dataset to another. Vocabulary transformation category mapping can also contain data transformations such as converting miles to kilometres or concatenation of a first name and a last name. Therefore, since vocabulary transformation category mappings consists of two or more RDF triples, and interlinks consist of a single triple - if the SCMR can sufficiently represent vocabulary transformation category mappings then it can sufficiently represent interlink category mappings.

In this experiment, there is potential bias as the second and third datasets (see below), used in this experiment, were created by the author of this thesis. The second dataset consists of SPARQL CONSTRUCT queries that are used to determine the ability of those queries with regard to representing vocabulary transformation category mappings. The third data consists of instance data collected from Linked Data datasets that the SPARQL CONSTRUCT (from second dataset) queries will be executed over to test their ability. Potential bias here is that these SPARQL CONSTRUCT queries do not
perform effectively or that only favourable instance data was collected for the third dataset. To alleviate any potential bias concerns, all data related to this experiment is made public and is published at the following web address\textsuperscript{10} and is available in the DVD that accompanies this thesis.

6.2.2. Hypotheses

As stated above, this experiment was performed in two parts and thus has two hypotheses. The first hypothesis is concerned with the testing of the SCMR use of SPARQL CONSTRUCT queries to express vocabulary transformation category mappings and is stated as follows:

- **Hypothesis H1**: SPARQL CONSTRUCT queries can be used to sufficiently express executable vocabulary transformation category mappings between Linked Data datasets.

To test if SPARQL CONSTRUCT queries can sufficiently express executable vocabulary transformation category mappings, their ability will be compared to that of the R2R Mapping Language (described in Chapter 3). The R2R Mapping Language was specifically designed to represent executable vocabulary transformation category mappings between Linked Data datasets and was chosen for comparison in this experiment per the discussion in Section 6.1 (under Experiment 1).

The second hypothesis is concerned with the testing of the SCMR use of the SPIN SPARQL syntax to encode SPARQL CONSTRUCT queries in RDF and is stated as follows:

- **Hypothesis H2**: The SPIN SPARQL syntax is sufficiently expressive to encode SPARQL CONSTRUCT query-based vocabulary transformation category mappings in RDF.

The second hypothesis was tested as, to the knowledge of the author of this thesis, no review of the expressivity of the SPIN SPARQL syntax with regard to representing SPARQL queries in RDF exists. Therefore, in order to gather evidence that the SPIN SPARQL syntax was suitable to use as part of the SCMR, this hypothesis tests its expressivity in regard to representing SPARQL CONSTRUCT query-based vocabulary transformation category mappings in RDF.

\textsuperscript{10} http://www.scss.tcd.ie/~meehanal/Experiment1/
6.2.3. Datasets

Three datasets were used in this experiment. The first is a dataset consisting of 72 vocabulary transformation category mappings between the DBpedia dataset and 11 other Linked Data datasets\(^{11}\). These mappings were created by the researchers of the R2R Framework (described in Chapter 3) to test their framework and are represented in the R2R Mapping Language. The creators of these mappings [Bi10] detail eight features that needed to be supported by the R2R Mapping Language in order to correctly represent the mappings. These features are listed as:

- Replacing URIs of a source dataset with URIs from a target dataset.
- Structural transformation of the RDF graph describing a data instance.
- Structural transformation of the RDF graph describing a data instance where an instance described by one RDF triple in one dataset results in the creation of multiple triples in another dataset.
- Literal value transformation using string functions (for example, concatenating first name and last name together to form a full name).
- Unit of measurement normalisation (for example, converting miles to kilometres or hour to minutes etc.)
- Datatype modifier applied to a literal value (for example, changing a literal from xsd:string to xsd:integer).
- Language modifier applied to a literal value (for example, changing or adding the ‘@en’ language tag to a literal).
- Creating a URI from (part of) a literal value or creating a literal value from (part of) a URI.

It is unknown to the author of this thesis if the list of features mentioned above is an exhaustive list of all features necessary to perform all possible vocabulary transformation category mapping problems that could arise. However, these mapping come from peer reviewed work, created independently of the research of this thesis and they consist of vocabulary transformation category mappings between real Linked Data datasets. Therefore, the author of this thesis believes that this mapping dataset is suitable for use to test if another mapping representation is sufficiently able to represent executable vocabulary transformation category mappings between Linked Data datasets. To provide further insight into the types of mappings within the dataset, they were

\(^{11}\) Misicbrainz, GeoNames, CIAFactbook, BookMashup, Project Gutenberg, USCensus, DailyMed, Diseasesome, LinkedMovieDataBase, DrugBank, Freebase.
categorised according to Scharffe’s correspondences patterns [Sc09]. The 72 mappings of the dataset span between three correspondence patterns, the: Equivalent Class, Equivalent Relation and Property Value Transformation patterns as illustrated in Figure 6-1 below. Of the 72 mappings, 15 fall under the Equivalent Class pattern, 57 fall under the Equivalent Relation pattern and 14 of the mappings under the Equivalent Relation pattern also fall under the Property Value Transformation pattern. Scharffe states that “Equivalent” correspondence patterns are the most common form of correspondences [Sc08]. The 72 mappings of the dataset fall between the Equivalent Class and Equivalent Relation patterns. This suggests that the mappings in this datasets are representative of mapping likely to be encountered out in the world and further increases the suitability of the dataset for use in the evaluations of this thesis.

![Figure 6-1. Breakdown of vocabulary transformation category mapping dataset into correspondence patterns.](image)

This dataset will be known as the **R2R vocabulary mapping dataset** from now on in this thesis.

The second dataset used in this experiment consists of the 72 mappings from the R2R vocabulary mapping dataset but each expressed as a SPARQL CONSTRUCT query instead of the R2R Mapping Language. This dataset was created by the author of this thesis and will be known as the **SP-CON vocabulary mapping dataset** in this thesis.

The third dataset used in this experiment consists of instance data from datasets that the mappings from the source of the R2R vocabulary mapping dataset reference\(^\text{12}\). This data was collected from SPARQL endpoints and data dump files of the datasets by the

\(^{12}\text{Note that no instance data related for the BookMashup dataset could be found. This meant that mappings related to the BookMashup dataset (two mappings) could not be used in this experiment. Out of the 72 mappings in the vocabulary mapping dataset, 70 of them were used.}\)
author of this thesis. This dataset was collected as it will serve as input for mappings so they can be executed and produce an output. The output of different approaches to represent and execute vocabulary transformation category mappings can then be compared to test for equivalent mapping functionality. This dataset will be known as the instance data dataset in this thesis.

6.2.4. Methodology
To test **H1**, which involves evaluating the expressivity of SPARQL CONSTRUCT queries in relation to expressing executable vocabulary transformation category mappings, it was compared to an existing approach to representing vocabulary transformation category mappings - the R2R Mapping Language. The approach undertaken to compare these two approaches is to execute both the R2R vocabulary mapping dataset and the SP-CON vocabulary mapping dataset over the instance data dataset. The outputs of the execution of these two sets of mappings are then compared to check if the outputs are identical for each mapping. If the outputs are identical, then the two approaches are equally as expressive at representing executable vocabulary transformation category mappings that appear in the datasets.

Testing **H2** evaluated the expressivity of the SPIN SPARQL syntax in regard to encoding SPARQL CONSTRUCT query-based vocabulary transformation category mappings in RDF. To do this, an official implementation of SPIN - TopBraid Composer 4.4.0 (free edition) was used. One feature of TopBraid Composer is it allows a user to input a SPARQL query and it will display the SPIN SPARQL syntax encoding of that query if possible to do so.

6.2.5. Procedure
This sub-section presents the procedures that were undertaken in the two parts of this experiment.

**Part One Procedure:**
The procedure for the first part of this experiment, concerned with testing **H1** is presented here. An Apache Jena triple-store with Fuseki SPARQL server version 1.3.0 was used in this experiment for the execution of the SP-CON vocabulary mapping dataset. 4GB of RAM was allotted to the triple store-store and SPARQL server and it was run on a machine with an Intel Core i7-3770 (3.40Ghz) processor.
1. The 70 individual mappings from the R2R vocabulary mapping dataset were executed (via the R2R Framework) over data from the instance data dataset. This execution resulted in 70 output files consisting of newly created triples.

2. The 70 individual mappings from the SP-CON vocabulary mapping dataset were manually executed (via a SPARQL processor) over the data from the instance data dataset that was loaded into a triple-store. The execution results of each mapping, which were newly created triples were recorded to a file, resulting in 70 output files.

3. The execution results of the respective mappings from the R2R vocabulary mapping dataset and the SP-CON vocabulary mapping dataset were manually compared, triple for triple, to see if they were identical.

**Part Two Procedure:**

The procedure for the second part of this experiment, concerned with testing H2 is as follows:

1. Each of the 70 mappings from the SP-CON vocabulary mapping dataset were manually input to the TopBraid Composer application to be encoded to RDF via the SPIN SPARQL syntax. An error is produced by TopBraid Composer if a query cannot be encoded via the SPIN SPARQL syntax.

**6.2.6. Results**

This sub-section presents the results obtained from the two parts of this experiment.

**Part One Results:**

For the first part of the experiment, it was found that execution results of the mappings from the R2R vocabulary mapping dataset and the SP-CON vocabulary mapping dataset were identical. Table 6-2 displays the number of identical and non-identical mapping execution results from both the R2R and SP-CON vocabulary mapping datasets.

| Table 6-2. Number of identical and non-identical execution results of mappings from the R2R and SP-CON vocabulary mapping datasets. |
|-------------|----------------|----------------|
| Identical Execution Results of Mappings from R2R and SP-CON Vocabulary Mapping Datasets | Non-Identical Execution Results of Mappings from R2R and SP-CON Vocabulary Mapping Datasets |
| Number      | 70             | 0              |

97
Part Two Results:
For the second part of the experiment, it was found that the SPIN SPARQL syntax is expressive enough to encode all 70 of the mappings from the SP-CON vocabulary mapping dataset in RDF. Table 6-3 displays the number of mappings from the SP-CON vocabulary mapping dataset that are able and not able to be encoded in RDF via the SPIN SPARQL syntax.

Table 6-3. Number of mappings able and not able to be encoded in RDF via the SPIN SPARQL syntax.

<table>
<thead>
<tr>
<th></th>
<th>Mappings Able to be Encoded in RDF via the SPIN SPARQL Syntax</th>
<th>Mappings Not Able to be Encoded in RDF via the SPIN SPARQL Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6-4 shows the average number of triples broken down per correspondence pattern of the mappings, when encoded in RDF via the SPIN SPARQL syntax and when represented in the R2R Mapping Language. The equivalent relation and the property value transformation correspondence patterns are grouped together as all instances of a property value transformation pattern occurred during an instance of an equivalent relation pattern.

Table 6-4. Average number of triples per mapping broken down by correspondence pattern.

<table>
<thead>
<tr>
<th>Correspondence Patterns</th>
<th>Equivalent Class</th>
<th>Equivalent Relation and Property Value Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average number of triples per mapping as SPIN</td>
<td>15</td>
<td>20.43</td>
</tr>
<tr>
<td>Average number of triples per mapping as R2R</td>
<td>3</td>
<td>3.32</td>
</tr>
</tbody>
</table>

6.2.7. Discussion
As can be seen from the results for the first part of this experiment, the execution results of the mappings from the R2R and SP-CON vocabulary mapping datasets are identical. This indicates that SPARQL CONSTRUCT queries are equally as expressive as the R2R Mapping Language for representing the types of vocabulary transformation category mappings that occur in the mapping dataset of the experiment. The R2R Mapping Language was specifically designed for representing and executing vocabulary transformation category mappings whereas SPARQL was not. Since SPARQL is a W3C recommendation and has been found to be equally as expressive as the as the R2R
Mapping Language - it would indicate that it is a suitable technology to be used in the SCMR to represent mappings. However, the drawback of this experiment is that the potential evaluation expressivity of both SPARQL CONSTRUCT queries and the R2R Mapping Language are limited by the mapping dataset itself. As noted above, it is unknown to the author if the features required to represent all these mappings are an exhaustive list. Therefore it is possible that a vocabulary transformation category mapping exists that would require support of additional features from a mapping representation in order to correctly represent it. In such a situation it is unknown if SPARQL CONSTRUCT queries have the sufficient expressivity.

For the second part of this experiment, the results show that the SPIN SPARQL syntax is expressive enough to encode all of the mappings from the SP-CON vocabulary mapping dataset in RDF. This indicates that for the types of vocabulary transformation category mappings that appear in the mapping dataset, the SPIN SPARQL syntax is able to encode them in RDF. This would indicate that the SPIN SPARQL syntax is suitable for use by the SCMR to encode mappings in RDF. However, the same drawback of the first part of the experiment applies to this part of the experiment.

The numbers in Table 6-4 show that the mappings represented as SPIN contain more triples than the mappings represented in the R2R Mapping Language. For mappings that fall under the equivalent class correspondence pattern, mappings represented in SPIN contain on average 5 times more triples than mappings represented in the R2R Mapping Language. For mappings that fall under the combination of the equivalent relation and property value translation correspondence patterns, the SPIN representation of mappings contain on average 6.15 times more triples than the R2R Mapping Language representation. The difference in the number of triples is due to the R2R Mapping Languages use of strings to represent mapping properties, whereas the SPIN SPARQL syntax does not use strings to represent any mapping properties. This results in the SPIN SPARQL syntax providing a more fine grain representation of a mapping over the R2R Mapping Language and is naturally less concise.

6.2.8. Conclusion
This section has presented the first experiment of this thesis and was concerned with evaluating the SCMR. The experiment consisted of two parts with two separate hypotheses. The evidence gathered from performing the first part of the experiment has
indicated that hypothesis \textbf{H1} is true with respect to the SP-CON vocabulary mapping dataset - SPARQL CONSTRUCT queries can be used to sufficiently represent executable vocabulary transformation category mappings between Linked Data datasets. From performing the second part of the experiment, there is evidence that hypothesis \textbf{H2} is true with respect to the SP-CON vocabulary mapping dataset.

This experiment has provided evidence that the design decisions to use SPARQL CONSTRUCT queries and the SPIN SPARQL syntax for the SCMR are good choices for a mapping representation. Indications are that SPARQL CONSTRUCT queries are as expressive at representing executable vocabulary transformation category mappings as the R2R Mapping Language. Moreover, an advantage SPARQL CONSTRUCT queries have over the R2R Mapping Language is that they are executable on any standard SPARQL processor. In addition, indications are that the SPIN SPARQL syntax will be able to represent all the SPARQL CONSTRUCT query-based vocabulary transformation category mappings in RDF. Mappings represented in SPIN are more fine grain than mappings represented in the R2R Mapping Language which does result in more triples per mapping.

The granularity of these two mapping representations are put to the test in the following experiment with regards to performing mapping maintenance and reuse tasks over them using SUMMR templates.

6.3. Experiment 2: Evaluation of SUMMR for Mapping Maintenance and Reuse Tasks 1 to 3

This section presents the second evaluation of the SUMMR methodology. \textit{Section 6.3.1} presents an introduction and motivation for the experiment. The experiment hypothesis is presented in \textit{Section 6.3.2}. The datasets that were used in this experiment are described in \textit{Section 6.3.3}. The mapping retrieval use cases used in this experiment are described in \textit{Section 6.3.4}. \textit{Section 6.3.5} describes the methodology undertaken for the experiment. The procedure taken is presented in \textit{Section 6.3.6}. Results and discussion are presented in \textit{Section 6.3.7} and \textit{Section 6.3.8} respectively. A conclusion is then presented in 6.3.9.

6.3.1. Introduction

In \textit{Chapter 3} (State of the Art) five tasks were derived that cover what is needed to effectively perform mapping maintenance and reuse.
This section presents a lab-based experiment that evaluates the ability of the SUMMR methodology query templates to support a potential user to perform mapping maintenance and reuse task numbers 1 to 3 (see Experiment 3 below for an evaluation of SUMMR’s support for tasks 4 and 5).

Recall from Chapter 5, that SUMMR supports mapping maintenance and reuse tasks by providing templates that use standard SPARQL queries. Thus, it is necessary to test that SPARQL, as a query language, can perform these tasks. To test this, a set of twelve mapping retrieval use cases (described further below), which encompass tasks 1 to 3, were derived from the research of Thomas et al. [Th11]. In this experiment, SPARQL is tested on its ability to effectively perform each of these twelve mapping retrieval use cases in four scenarios. In the first and second scenarios, bespoke SPARQL queries were designed (by the author of this thesis) to perform the use cases over mappings represented in SCMR and mappings represented in the R2R Mapping language respectively. In the third scenario, SUMMR templates, specialised for SCMR are used to perform the use cases over mappings represented in SCMR. In the fourth scenario, SUMMR templates, specialised for the R2R Mapping Language are used to perform the use cases over mappings represented in the R2R Mapping Language. The bespoke queries were created to show which use cases SPARQL queries were capable of performing when not being in template form. Performing the use cases with bespoke queries, then using SUMMR templates specialised for SCMR and specialised for the R2R Mapping Language was done to highlight any constraints that may be imposed on the queries by being in template form. Performing the use cases over mappings represented in SCMR and the R2R Mapping Language will highlight which representation affords a greater number of mapping maintenance and reuse tasks to be performed.

In this experiment, the author is aware there is potential for bias. This bias is centred on the bespoke queries and the SUMMR templates designed to perform the mapping retrieval use cases over mappings represented in SCMR and the R2R Mapping Language. For example the queries designed to be performed over the SCMR could be tweaked to produce more favourable results as those designed to be performed over the R2R Mapping Language. To alleviate bias concerns, all data related to this experiment (including bespoke queries, SUMMR templates, datasets, gold standards) are made public and published online at the following web address\(^\text{13}\) and is available in the DVD.

\(^{13}\) http://www.scss.tcd.ie/~meehanal/Experiment2b/
that accompanies this thesis. In addition, the gold standard dataset used in this experiment was verified by a postdoctoral researcher in the Knowledge and Data Engineering Group in Trinity College Dublin.

6.3.2. Hypotheses
The first hypothesis for this experiment is as follows:

- **Hypothesis H3**: The SCMR affords a greater number of mapping maintenance and reuse tasks to be performed over it compared to the R2R Mapping Language.

The second hypothesis is stated as follows:

- **Hypothesis H4**: The effectiveness of SUMMR templates is equivalent to the bespoke queries (created in this experiment) with regard to performing the twelve mapping retrieval use cases used in this experiment.

The effectiveness metric stated in the hypothesis for this experiment is broken down into *Recall* and *Precision*. Recall and Precision are used as this experiment as SUMMR templates are involved in an information retrieval task - where correctly detected invalid mappings are being retrieved from a set of mappings. For this it is important to correctly detect and retrieve the invalid mappings (Recall) while withholding still valid mappings (Precision).

6.3.3. Datasets
Three datasets were used in this experiment. The first is the R2R vocabulary mapping dataset (also used in the first experiment). The mappings in this dataset have been given an identification number from 1 to 72.

The second dataset is an extension to the SP-CON vocabulary mapping datasets (as used in the first experiment). The mappings in the SP-CON vocabulary mapping dataset have been extended to be complete SCMRs as per the example in Figure 5-3 (in Chapter 5). The SCMR representations in this dataset are annotated with the same meta-data as the mappings from the R2R vocabulary mapping dataset and have been given an identification number from 1 to 72. This dataset was created by the author of this thesis and will be known as the **SCMR vocabulary mapping dataset** in this thesis.

The third dataset used is a gold standard dataset that contains the correct results to the twelve concrete experiment 2 mapping retrieval use cases (described below). The results to these use cases are particular mappings and this dataset contains the mapping...
identifier number of the correct mappings for each use case. This dataset was created by the author of this thesis and will be known as the **mapping retrieval use case gold standard dataset**. The **mapping retrieval use case gold standard dataset** was validated by a postdoctoral researcher with 7 years of experience in Semantic Web and Linked Data research at the Knowledge and Data Engineering Group in Trinity College Dublin. This validation was achieved through manually checking each of the mappings, from the R2R vocabulary mapping dataset, for each of the twelve concrete experiment 2 mapping retrieval use cases, to see which mappings adhere to each use case.

### 6.3.4. Mapping Retrieval Use Cases

The mapping retrieval use cases used in this experiment are derived from mapping retrieval use cases introduced and motivated in the research of Thomas et al. [Th11]. In their research, they introduce 16 mapping retrieval use case which are presented in **Table 6-5** below. Many of these use cases are concerned with finding mappings with annotated meta-data which describe specific properties of the mappings such as whether it was created automatically, manually or in a combination and the specific matching algorithm configuration. The two mapping datasets used in this experiment (the R2R vocabulary mapping dataset and the SCMR vocabulary mapping dataset) do not have this meta-data and therefore not all of the use cases are applicable for use in this experiment.

**Table 6-5. Mapping retrieval use cases from Thomas et al [Th11].**

<table>
<thead>
<tr>
<th>ID</th>
<th>Mapping Retrieval Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_1</td>
<td>Find mappings between specific source and target ontologies</td>
</tr>
<tr>
<td>U_2</td>
<td>Find mappings between specific source and target ontology elements</td>
</tr>
<tr>
<td>U_3</td>
<td>Find mappings for a particular ontology</td>
</tr>
<tr>
<td>U_4</td>
<td>Find mappings for a particular ontology element</td>
</tr>
<tr>
<td>U_5</td>
<td>Find mappings between a specific source ontology to any target ontology with specified characteristics</td>
</tr>
<tr>
<td>U_6</td>
<td>Find mappings expressed in particular mapping format</td>
</tr>
<tr>
<td>U_7</td>
<td>Find mappings for specific correspondence types</td>
</tr>
<tr>
<td>U_8</td>
<td>Find mappings created either automated, manually or in a combination</td>
</tr>
<tr>
<td>U_9</td>
<td>Find mappings created by a specific matching type</td>
</tr>
<tr>
<td>U_10</td>
<td>Find mappings created by a specific matching algorithm implementation</td>
</tr>
<tr>
<td>U_11</td>
<td>Find mappings created by a given matching algorithm configuration</td>
</tr>
<tr>
<td>U_12</td>
<td>Find automated created mappings based on matches with a high confidence level</td>
</tr>
<tr>
<td>U_13</td>
<td>Find manually created mappings depending on the involved users</td>
</tr>
<tr>
<td>U_14</td>
<td>Find mappings created by a particular author</td>
</tr>
<tr>
<td>U_15</td>
<td>Find a specific version of a mapping</td>
</tr>
<tr>
<td>U_16</td>
<td>Find mappings created for a particular context</td>
</tr>
</tbody>
</table>
To find which use cases would be applicable to test with the datasets used in this experiment, the author of this thesis analysed the mapping datasets, the mapping retrieval use cases and the mapping maintenance and reuse tasks 1 to 3. It was determined that mapping retrieval use cases U_1, U_2, U_3, U_4, U_5, U_14 and U_15 could be used in this experiment as explained below:

- Mapping retrieval use cases U_1 to U_4 are based on discovering and retrieving mappings based on specific search criteria related to the source and target properties of a mapping. The use cases vary based on how general or specific the search criteria is. For example, use cases U_1 and U_3 could be classed as specific and general versions of each other. These use cases are suitable for use as the source and target mapping properties are available in the mapping datasets used in this experiment. These four use cases also encompass mapping maintenance and reuse tasks 1 (Discover a mapping based on search criteria) and 3 (Retrieve the entirety of a mapping).

- Mapping retrieval use case U_5 is concerned with discovering and retrieving mappings based on specific mapping source criteria and mappings with a specific characteristic. This use case contains a combination of finding mappings based on source search criteria (or target search criteria could be used) and some characteristic of a mapping. Vocabulary transformation category mappings have characteristics such as containing data transformations. This use case is suitable for use as it details finding mappings based on a combination of different types of search criteria and characteristics such as data transformations - which are available in the mapping datasets used in this experiment. This use cases encompass mapping maintenance and reuse tasks 1 and 3.

- Mapping retrieval use case U_14 is concerned with discovering and retrieving mappings created by a particular author. This use case is suitable for use as it is concerned with finding mappings based on a meta-data term. Mappings in the datasets used in this experiment have meta-data terms and specifically have a meta-data term for the author of the mapping. This use cases encompass mapping maintenance and reuse tasks 1 and 3.

- Mapping retrieval use case U_15 is concerned with finding the version of a specific mapping. This use case is different from the rest as it is not concerned with finding mappings based on specific criteria, it is based on finding a property of a specific mapping. While the mappings in the datasets used in this
experiment do not contain a version, the premise behind this use case can apply
to other properties of a mapping, such as the source and target properties of a
mapping. This is the only use case which encompasses mapping maintenance
and reuse tasks 2 (Discover and retrieve individual properties of a mapping).

After the use cases above were determined suitable for use, twelve use cases were
derived for use in this experiment. They are named experiment 2 mapping retrieval use
cases and were derived from mapping retrieval use cases U_1, U_2, U_3, U_4, U_5,
U_14 and U_15. The experiment 2 mapping retrieval use cases are presented in Table 6-
6 on the following page, which also shows which of the mapping retrieval use cases
each they were derived from. Experiment 2 mapping retrieval use cases 10, 11 and 12
were each derived from combining two of the mapping retrieval use cases. These were
derived to provide more challenging use cases to be performed.

With the creation of the experiment 2 mapping retrieval use cases, a set of twelve
concrete experiment 2 mapping retrieval use cases were created which specify exactly
the mappings that are to be retrieved. It is these concrete use cases that are performed in
the experiment. They are presented in Table 6-7 on the following page, which also
shows which of the mapping maintenance and reuse tasks are encompassed in the use
case.

<table>
<thead>
<tr>
<th>ID</th>
<th>Experiment 2 Mapping Retrieval Use Case</th>
<th>Use Case Derived From</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_1</td>
<td>Find all mappings between specific source and target vocabularies</td>
<td>U_1</td>
</tr>
<tr>
<td>E_2</td>
<td>Find all mappings between specific source and target vocabulary terms</td>
<td>U_2</td>
</tr>
<tr>
<td>E_3</td>
<td>Find all mappings for a particular vocabulary</td>
<td>U_3</td>
</tr>
<tr>
<td>E_4</td>
<td>Find all mapping for a particular vocabulary term</td>
<td>U_4</td>
</tr>
<tr>
<td>E_5</td>
<td>Find all mappings that contain a data transformation of some kind</td>
<td>U_5</td>
</tr>
<tr>
<td>E_6</td>
<td>Find all mapping that contain a specific data transformation</td>
<td>U_5</td>
</tr>
<tr>
<td>E_7</td>
<td>Find the individual source and target terms of a specific mapping</td>
<td>U_15</td>
</tr>
<tr>
<td>E_8</td>
<td>Find all mappings created on a specific date</td>
<td>U_14</td>
</tr>
<tr>
<td>E_9</td>
<td>Find all mappings created by a specific person</td>
<td>U_14</td>
</tr>
<tr>
<td>E_10</td>
<td>Find all mappings for a particular vocabulary that contain a transformation</td>
<td>U_3, U_5</td>
</tr>
<tr>
<td>E_11</td>
<td>Find the individual source and target terms of all mappings that contain a specific transformation</td>
<td>U_5, U_15</td>
</tr>
<tr>
<td>E_12</td>
<td>Find all mappings between specific source and target vocabulary terms, created by a specific person on a specific date</td>
<td>U_2, U_14</td>
</tr>
<tr>
<td>ID</td>
<td>Concrete Experiment 2 Mapping Retrieval Use Case</td>
<td>Tasks Encompassed</td>
</tr>
<tr>
<td>----</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>C_1</td>
<td>Find all mappings that use a ‘DBpedia’ term as a source and a ‘Geonames’ term as a target.</td>
<td>1, 3</td>
</tr>
<tr>
<td>C_2</td>
<td>Find all mappings that have the ‘Factbook Country’ term as the source and the ‘DBpedia Country’ term as the target.</td>
<td>1, 3</td>
</tr>
<tr>
<td>C_3</td>
<td>Find all mappings for the “Linked Movie Database (LMDB)” vocabulary.</td>
<td>1, 3</td>
</tr>
<tr>
<td>C_4</td>
<td>Find all mappings that use the ‘foaf name’ term.</td>
<td>1, 3</td>
</tr>
<tr>
<td>C_5</td>
<td>Find all mappings that contain a data transformation.</td>
<td>1, 3</td>
</tr>
<tr>
<td>C_6</td>
<td>Find all mappings that contain a ‘divide’ transformation.</td>
<td>1, 3</td>
</tr>
<tr>
<td>C_7</td>
<td>Find the individual source and target terms of the ‘freebaseToDBpediaInstrumentProperty’ mapping.</td>
<td>2</td>
</tr>
<tr>
<td>C_8</td>
<td>Find all mappings created on ‘2010-07-03’.</td>
<td>1, 3</td>
</tr>
<tr>
<td>C_9</td>
<td>Find all mappings created by '<a href="http://www4.wiwiss.fu-berlin.de/is-group/resource/persons/Person30">http://www4.wiwiss.fu-berlin.de/is-group/resource/persons/Person30</a>'.</td>
<td>1, 3</td>
</tr>
<tr>
<td>C_10</td>
<td>Find all mappings for the ‘DBpedia’ vocabulary that contain a data transformation.</td>
<td>1, 3</td>
</tr>
<tr>
<td>C_11</td>
<td>Find the individual source and target terms of all mappings that contain a ‘multiply’ data transformation.</td>
<td>1, 2</td>
</tr>
<tr>
<td>C_12</td>
<td>Find all mappings between the ‘DBpedia areaWater’ term as the source, ‘UScensus waterArea’ term as the target, created by '<a href="http://www4.wiwiss.fu-berlin.de/is-group/resource/persons/Person30">http://www4.wiwiss.fu-berlin.de/is-group/resource/persons/Person30</a>' on ‘2010-06-22’</td>
<td>1, 3</td>
</tr>
</tbody>
</table>

6.3.5. Methodology

Four sets of queries were used. The first two sets of queries consisted of twelve bespoke SPARQL queries each. The third set of queries consisted of SUMMR templates (combinations of SUMMR task templates 1-3) specialised for SCMR. The fourth set of queries consisted of SUMMR templates (combinations of SUMMR task templates 1-3) specialised for the R2R Mapping Language.

The bespoke queries were created by the author of this thesis to perform the concrete experiment 2 mapping retrieval use cases over mappings represented in SCMR and the R2R Mapping language. Care was taken during creation to ensure they could effectively perform each use case.

The SUMMR templates specialised for SCMR and the R2R Mapping Language were created by the author of this thesis. In the same manner as the bespoke queries, care was taken during their creation to ensure they could effectively perform each use case.

To test the hypotheses for this experiment, the two sets of bespoke queries, and the two sets of SUMMR templates specialisations were executed against their respective
mapping dataset. The execution result of each query was recorded and compared to the mapping retrieval use case gold standard dataset, and the metrics of precision and recall were used to determine their effectiveness.

While not tested in the hypotheses for this experiment, the execution time of each query was also recorded as a measure of relative efficiency of each query. This was done to provide insight into the difference in efficiency of queries in a bespoke form compared to queries in template form.

6.3.6. Procedure

The procedure that was undertaken in this experiment is described in this sub-section. An Apache Jena triple-store with Fuseki SPARQL server version 1.3.0 was used in this experiment for the execution of the queries. 4GB of RAM was allotted to the triple store-store and SPARQL server and it was run on a machine with an Intel Core i7-3770 (3.40Ghz) processor. The procedure was as follows:

1. The SCMR vocabulary mapping dataset and the R2R vocabulary mapping dataset were loaded into separate named graphs in the triple-store.
2. The bespoke query set to perform the 12 use cases over mappings represented in SCMR were executed one-by-one. The queries were executed five times each to provide a greater a wider sampling range for the query execution time. No notable variation in the execution time was observed for the multiple execution of the queries. For each query and execution, the execution result and the execution time were recorded.
3. The bespoke query set to perform the 12 use cases over mappings represented in the R2R Mapping Language were executed one-by-one five times each. Again, no notable variation in the execution time was observed for the multiple execution of the queries. For each query and execution, the execution result and the execution time were recorded.
4. The SUMMR templates specialised for SCMR were executed one-by-one, five times each for the 12 use cases (over the SCMR vocabulary mapping dataset). No notable variation in the execution time was observed for the multiple execution of the queries. For each query and execution, the execution result and the execution time were recorded.
5. The SUMMR templates specialised for the R2R Mapping Language were executed one-by-one, five times each for the 12 use cases (over the R2R
vocabulary mapping dataset). Similar to the previous cases, no notable variation in the execution time was observed for the multiple execution of the queries. For each query and execution, the execution result and the execution time were recorded.

6. All the execution results for each of the 12 use cases were compared to the mapping retrieval gold standard dataset.

6.3.7. Results
This sub-section presents the results obtained from performing the experiment.

Recall and precision metrics were used to determine the effectiveness of each query (in bespoke form or template form) for performing each concrete experiment 2 mapping retrieval use case. The recall of a query was determined by comparing the query execution results to the mapping retrieval use case gold standard dataset to ensure that all the gold standard results appear within the query execution results. Determining the precision of a query is similar to the recall, but this time the query execution results were compared to the gold standard results to ensure that no results other than the gold standard results are present within the query results. The recall and precision results for bespoke queries and the templates are displayed in Table 6-8.

Table 6-8. Recall (R) and Precision (P) results of the bespoke queries and SUMMR templates performed over mappings represented in SCMR and the R2R Mapping Language for the 12 concrete mapping retrieval use cases.

<table>
<thead>
<tr>
<th>Concrete Use Case ID</th>
<th>Bespoke Queries</th>
<th>SUMMR Templates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SCMR R P</td>
<td>R P</td>
</tr>
<tr>
<td>C_1</td>
<td>1.0 1.0</td>
<td>1.0 1.0</td>
</tr>
<tr>
<td>C_2</td>
<td>1.0 1.0</td>
<td>1.0 1.0</td>
</tr>
<tr>
<td>C_3</td>
<td>1.0 1.0</td>
<td>1.0 1.0</td>
</tr>
<tr>
<td>C_4</td>
<td>1.0 1.0</td>
<td>1.0 1.0</td>
</tr>
<tr>
<td>C_5</td>
<td>1.0 1.0</td>
<td>1.0 1.0</td>
</tr>
<tr>
<td>C_6</td>
<td>1.0 1.0</td>
<td>1.0 1.0</td>
</tr>
<tr>
<td>C_7</td>
<td>1.0 1.0</td>
<td>0.0 1.0</td>
</tr>
<tr>
<td>C_8</td>
<td>1.0 1.0</td>
<td>1.0 1.0</td>
</tr>
<tr>
<td>C_9</td>
<td>1.0 1.0</td>
<td>1.0 1.0</td>
</tr>
<tr>
<td>C_10</td>
<td>1.0 1.0</td>
<td>1.0 1.0</td>
</tr>
<tr>
<td>C_11</td>
<td>1.0 1.0</td>
<td>0.0 0.0</td>
</tr>
<tr>
<td>C_12</td>
<td>1.0 1.0</td>
<td>1.0 1.0</td>
</tr>
</tbody>
</table>
The execution time for each query was calculated by taking the mean of the 5 runs for each query. Table 6.9 displays the mean query execution time of the bespoke queries and the templates.

Table 6.9. Mean execution time of the bespoke queries and templates performed over mappings represented in SCMR and the R2R Mapping Language for the 12 concrete use cases.

<table>
<thead>
<tr>
<th>Concrete Use Case ID</th>
<th>SCMR Mean Execution Time (ms)</th>
<th>R2R Mean Execution Time (ms)</th>
<th>SCMR Mean Execution Time (ms)</th>
<th>R2R Mean Execution Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_1</td>
<td>326</td>
<td>239</td>
<td>346</td>
<td>244</td>
</tr>
<tr>
<td>C_2</td>
<td>206</td>
<td>273</td>
<td>307</td>
<td>235</td>
</tr>
<tr>
<td>C_3</td>
<td>286</td>
<td>251</td>
<td>394</td>
<td>251</td>
</tr>
<tr>
<td>C_4</td>
<td>218</td>
<td>245</td>
<td>359</td>
<td>245</td>
</tr>
<tr>
<td>C_5</td>
<td>233</td>
<td>206</td>
<td>379</td>
<td>203</td>
</tr>
<tr>
<td>C_6</td>
<td>218</td>
<td>225</td>
<td>377</td>
<td>227</td>
</tr>
<tr>
<td>C_7</td>
<td>237</td>
<td>-</td>
<td>238</td>
<td>-</td>
</tr>
<tr>
<td>C_8</td>
<td>208</td>
<td>210</td>
<td>215</td>
<td>213</td>
</tr>
<tr>
<td>C_9</td>
<td>215</td>
<td>215</td>
<td>225</td>
<td>223</td>
</tr>
<tr>
<td>C_10</td>
<td>321</td>
<td>237</td>
<td>465</td>
<td>252</td>
</tr>
<tr>
<td>C_11</td>
<td>252</td>
<td>-</td>
<td>464</td>
<td>-</td>
</tr>
<tr>
<td>C_12</td>
<td>234</td>
<td>258</td>
<td>287</td>
<td>241</td>
</tr>
</tbody>
</table>

6.3.8. Discussion

The results in Table 6-8 show that the bespoke queries for mappings represented in SCMR achieved a recall and precision of 1.0 for all 12 of the use. The bespoke queries for mappings represented in the R2R Mapping Language achieved a recall and precision of 1.0 for all use cases except C_7 and C_11. Use cases C_7 and C_11 contain mapping maintenance and reuse task 2 - “Discover and retrieve individual properties of a mapping”. It was found that SPARQL queries could not be created to achieve this for mappings represented in the R2R Mapping Language. This is caused by how the R2R Mapping Language represents some of its mapping properties (such the source, target and data transformation properties) as a string. Accessing these properties individually is difficult and requires complex string functions. In order to return the individual source and target terms of an R2R mapping, a string explode function would be required, which SPARQL 1.1 does not incorporate. The problem would be compounded further when there are multiple triple patterns in the R2R source or target patterns, as each triple pattern would first have to be separated and then the individual elements
from each pattern would have to be separated. This problem could be handled using extensions to SPARQL or custom code but that defeats the purpose of designing templates based on the SPARQL standard.

The results in Table 6-8 also show that SUMMR templates specialised for SCMR achieved a recall and precision of 1.0 for all 12 of the use. SUMMR templates specialised for the R2R Mapping Language achieved the same results as the bespoke queries as they achieved a recall and precision of 1.0 for all use cases except C_7 and C_11. The SUMMR templates specialised for the R2R Mapping Language could not perform these use cases due to the same reasons that the bespoke queries could not.

The results show that mapping maintenance and reuse task 2 could not be performed by the bespoke queries and the SUMMR templates specialised for the R2R Mapping Language over mappings represented in the R2R Mapping Language. However it could be performed by the bespoke queries and SUMMR templates specialised for SCMR over mappings represented in SCMR. This indicates that the SCMR may afford greater mapping maintenance and reuse tasks to be performed over it compared to the R2R Mapping Language for the mappings in the datasets used in this experiment. The results also show that the precision and recall of the bespoke queries and SUMMR templates are identical for each of the 12 use cases. This indicates that SUMMR templates are as effective as the bespoke templates meaning that the effectiveness of SUMMR templates are not hindered by being in template form.

The query execution times presented in Table 6-9 indicate that the execution time of SUMMR templates are slower than the bespoke queries for each of the 12 use cases with the largest difference in execution time is 212ms. This is expected as the bespoke queries were designed to be as efficient as possible for each specific use. While the bespoke queries executed quicker, a drawback is that time must be spent on their creation and tuning them to be efficient (roughly 5-15 minutes spent on each bespoke query for this.), whereas the templates come ready to use to a large extent.

The SUMMR templates specialised for SCMR execution times are also slower than the templates specialised for the R2R Mapping Language, with the largest difference in execution time is 213ms. This is also expected as the SCMR is more fine grain than the R2R Mapping Language. Therefore, SUMMR templates (designed to be performed over mappings represented in SCMR) require more triple patterns in the queries to access a mappings properties at the fine granularity. The author of this thesis feels that even for the two cases where the biggest difference in execution times are noted, this
difference is not so unreasonable as to prefer one set of queries over the other in terms of query speed. Overall the results indicate that queries designed to perform maintenance and reuse tasks over mappings represented in the R2R Mapping Language are quicker than queries designed to be performed over mappings represented in SCMR. However, while these queries may be quicker, they are not as effective.

6.3.9. Conclusion

This section has presented the second experiment of this thesis and was concerned with evaluating SUMMR templates to perform mapping maintenance and reuse tasks 1 to 3 over mappings represented in SCMR.

The evidence gathered from performing the experiment has shown that the first hypothesis (H3) is true - the SCMR affords for more mapping maintenance and reuse tasks to be performed over it compared to the R2R Mapping Language. This was shown by the bespoke queries and the SUMMR templates specialised for the R2R Mapping Language inability to perform use cases C_7 and C_11. Evidence from the experiment has also shown that the second hypothesis (H4) is also true - the effectiveness of SUMMR templates are not hindered by being in template form.

This experiment has provided evidence that SUMMR templates specialised for SCMR can effectively perform mapping maintenance and reuse tasks 1-3 over mappings represented in SCMR. It has shown that different mapping representations afford a greater number of mapping maintenance and reuse tasks to be performed when using SUMMR templates. Mapping maintenance and reuse task 2 is of particular importance to perform as it is the second most involved task used in the six mapping maintenance and reuse use cases (see Table 5-2 in Chapter 5).

6.4. Experiment 3: Evaluation of SUMMR to Perform Mapping Maintenance and Reuse Tasks 4 and 5

The Third evaluation of the SUMMR methodology is presented in this section. An introduction to the experiment is presented in Section 6.4.1. The hypotheses are presented in Section 6.4.2. A description of the datasets used in this experiment is presented in Section 6.4.3. Section 6.4.4 describes the methodology undertaken for the experiment. The procedure is described in Section 6.4.5. The results are presented in Section 6.4.6. A discussion is presented in Section 6.4.7 and conclusion is presented in Section 6.4.8.
6.4.1. Introduction

This section presents the third experiment in this thesis. It is a lab-based experiment that evaluates the ability of the SUMMR Query Templates to support a potential user to perform mapping maintenance and reuse tasks numbers 4 and 5. Hence it complements Experiment 2 which focused on tasks 1 to 3. This involved testing SUMMR Use Case Templates 5 and 6 (which encompass tasks 4 and 5) in two parts. The first part of the experiment is concerned with testing SUMMR Use Case Template 5 through using it to discover invalid mappings from a set of vocabulary transformation and a set of interlink category mappings, both represented in SCMR. The second part of the experiment is concerned with testing SUMMR Use Case Template 6 by using it to repair both vocabulary transformation interlink category mappings found to be invalid for the first part of the experiment. Note that the R2R Mapping Language is not used in this experiment, as in experiment 2 it was shown that mapping maintenance and reuse task 2 could not be performed over the R2R Mapping Language using SUMMR templates. Since mapping maintenance and reuse use case 5 involves task 2 and use case 5 is used in the experiment, SUMMR templates specialised for the R2R Mapping Language would fail to perform use case 5.

In experiment 3, there is a possibility for potential bias centred on the SUMMR templates to discover invalid mappings and the templates to repair invalid mappings, as these templates were created by the author. Other potential bias is around the gold standard datasets used to evaluate the effectiveness of the SUMMR templates. To alleviate these potential bias concerns, all data related to this experiment are published online at the following web address14 and is available in the DVD that accompanies this thesis. The gold standard datasets used in this experiment were created by a postdoctoral researcher in the Knowledge and Data Engineering Group in Trinity College Dublin.

6.4.2. Hypotheses

There are two hypotheses for this experiment. The first hypothesis is concerned with the testing of SUMMR Use Case Template 5 and is stated as follows:

- **Hypothesis H5**: SUMMR Use Case Template 5 specialised for SCMR will achieve a recall and precision of 1.0 in regards to discovering correctly invalid vocabulary transformation category mappings and interlink

14 http://www.scss.tcd.ie/~meehanal/Experiment3/
category mappings from the SCMR vocabulary mapping dataset and the SCMR interlink dataset respectively.

The second hypothesis is concerned with the testing of SUMMR Use Case Template 6 and is stated as follows:

- **Hypothesis H6**: SUMMR Use Case Template 6 specialised for SCMR will repair 100% of the invalid vocabulary transformation and interlink category mappings discovered from the first part of the experiment.

### 6.4.3. Datasets

Four datasets were used in this experiment. The first two datasets consist of vocabulary transformation category mappings and interlink category mappings. The second two datasets are gold standard datasets. The gold standard datasets were created by a postdoctoral researcher with 7 years of experience in Semantic Web and Linked Data research at the Knowledge and Data Engineering Group in Trinity College Dublin.

The first dataset is the SCMR vocabulary mapping dataset as used and described in experiment two.

The second dataset consists of an 43 interlink category mappings between instances in the DBpedia and the DailyMed datasets, which use the `owl:sameAs` property. These instances consist of pharmaceutical companies and drugs. These 43 interlink category mappings are the entire set of mappings between the DBpedia and DailyMed datasets for the DBpedia v.3.3 release. For this experiment, these 43 interlink category mappings were represented in SCMR by the author of this thesis and the mappings were numbered 1 to 43. This dataset will be known as the **SCMR interlink dataset** from now on in the thesis.

The third dataset, is a gold standard dataset that contains the mappings from the SCMR vocabulary mapping dataset and the SCMR interlink dataset that are classed as syntactically invalid (as described in Section 5.6). The invalid vocabulary transformation category mappings were determined through ensuring that the DBpedia vocabulary terms from the mappings in the SCMR vocabulary mapping dataset, appear exactly as they are, in version 3.3 of the DBpedia vocabulary. If a vocabulary term does not appear exactly as is, then that vocabulary term and any mappings that use it were marked as invalid. For each invalid vocabulary term found, an updated or equivalent substitute term was searched for (with the aid of SPARQL queries) in version 3.3 of the DBpedia vocabulary that will be used to repair the invalid mapping (where a substitute
term was found). The invalid interlink category mappings were determined through checking for syntactic changes in instance URIs. This was done through ensuring that the DBpedia and DailyMed instance URIs from the mappings in the SCMR interlink dataset, exist exactly as they are, in the DBpedia v.2015-04 dataset release and a dump of the DailyMed dataset (that was collected via the DailyMed SPARQL endpoint on 02-September-2015) respectively. If not, that instance URI and any mappings that use it were marked as invalid. For each invalid instance URI found, an updated or equivalent substitute instance URI was searched for (with the aid of SPARQL queries) in the DBpedia v.2015-04 dataset release and the dump of the DailyMed dataset that will be used to repair the invalid interlink category mapping (where a substitute instance URI was found). This third dataset will be known as the invalid mapping gold standard dataset from now on in the thesis.

The fourth dataset, which is another gold standard dataset, was created after the invalid mapping gold standard dataset was created. Each of the invalid vocabulary transformation and interlink category mappings found were manually repaired - where it was possible to do so with a substitute vocabulary term or instance URI respectively. Care was taken to ensure the mappings were repaired correctly. For each of the repaired vocabulary transformation category mappings, instance data related to these mappings was collected from respective datasets the mappings reference. Each repaired vocabulary transformation category mapping was executed against its respective collected instance data and the results were recorded. Each repaired interlink category mapping was executed and the results were recorded. If a repaired mapping executed correctly - this indicates that a mapping was correctly repaired. The SCMR interlinks do not need instance data to execute against to produce a result. This dataset will be known as the mapping execution results gold standard dataset from now on in this thesis.

6.4.4. Methodology
Testing hypothesis H5 involves testing SUMMR Use Case Template 5 to correctly discover invalid vocabulary transformation and interlink category mappings from the SCMR vocabulary mapping dataset and the SCMR interlink dataset respectively, compared to the invalid mapping gold standard dataset.

To discover invalid vocabulary transformation category mappings, a SUMMR Use Case Template 5 specialised for SCMR was established to compare DBpedia vocabulary terms from the mappings in the SCMR vocabulary mapping dataset, to the
terms in the DBpedia v.3.3 vocabulary. By doing this, the template checks if the vocabulary terms from the mappings exist exactly as they are in the DBpedia v3.3 vocabulary. If the vocabulary terms from the mappings do not exist in the DBpedia v.3.3 vocabulary, then those terms and hence, the mappings that contain those terms are classed as invalid in relation the DBpedia v.3.3 vocabulary.

To discover invalid interlink category mappings, another SUMMR Use Case Template 5 specialised for SCMR was established to compare the DBpedia and the DailyMed instance URIs from the mappings in the SCMR interlink dataset, to the instance URIs in the DBpedia v.2015-04 dataset and the DailyMed dataset respectively. By doing this, the template finds if the DBpedia and the DailyMed instance URIs from the mappings exist exactly as they are in the DBpedia and DailyMed datasets respectively. If a DBpedia or DailyMed instance URI does not exist in the DBpedia v.2015-04 dataset and the DailyMed dataset, then those instance URIs and the mappings that contain them are classed as invalid.

The execution results of each of the two SUMMR Use Case 5 templates (one to discover invalid vocabulary transformation category mappings and one to discover invalid interlink category mappings) return mappings that have been discovered as invalid. To test the capabilities of the templates, the execution results were compared to the invalid mapping gold standard and the metrics of precision and recall were used to determine their effectiveness.

Testing hypothesis **H6** involves testing SUMMR Use Case Template 6 in regard to correctly repairing invalid vocabulary transformation category mappings and invalid interlink category mappings. To test this, SUMMR use case template 6 was used to repair the invalid vocabulary transformation s and the invalid interlink category mapping discovered in the first part of this experiment. During the creation of the invalid mapping gold standard dataset, new vocabulary transformation category terms and instance URIs were found that could be used to repair these invalid mappings. Using these new vocabulary terms and instance URIs with SUMMR Use Case Template 6, the templates were executed which updated and repaired the SCMR representations of the mappings. To ensure that a mapping was correctly repaired, the newly repaired mappings were executed and the execution results were recorded. The reasoning behind this is that an incorrectly repaired mappings would either not execute at all, or would produce wrong results. The execution results of the repaired mappings were then
compared to the mapping execution results gold standard dataset to ensure the execution results were identical and thus confirming a mapping was correctly repaired.

6.4.5. Procedure

This sub-section describes the procedures that was undertaken in the two parts of this experiment. An Apache Jena triple-store with Fuseki SPARQL server version 1.3.0 was used in this experiment for the execution of SUMMR Use Case Templates 5 and 6. 4GB of RAM was allotted to the triple store-store and SPARQL server and it was run on a machine with an Intel Core i7-3770 (3.40Ghz) processor.

**Part One Procedure:**

The procedure for the first part of this experiment, which was concerned with discovering invalid mappings, is as follows:

1. The SCMR vocabulary mapping dataset was loaded into a named graph in the triple-store.
2. Version 3.3 of the DBpedia vocabulary was loaded into another named graph in the triple-store.
3. SUMMR Use Case Template 5 specialised for SCMR was used for the detection of invalid vocabulary transformation category mappings. In the template, the named graph URIs of the SCMR vocabulary mapping dataset and version 3.3 of the DBpedia vocabulary were specified for the $NAMED_GRAPH placeholders in the template. The template was then executed and the invalid vocabulary transformation category mappings detected were recorded and stored in a text file.
4. The SCMR interlink dataset was loaded into a third named graph in the triple-store.
5. The DailyMed dump file was loaded into a fourth named graph in the triple-store.
6. SUMMR Use Case Template number 5 specialised for SCMR was then used to detect invalid interlink category mappings from the SCMR interlink dataset. In the template, the named graph URIs of the SCMR interlink dataset and the DailyMed dump were specified for the $NAMED_GRAPH placeholders in the template. A federated SPARQL call to the DBpedia SPARQL endpoint was also specified for the $REMOTE_SPARQL_ENDPOINT placeholder in
the template. The completed query template was then executed and the results of the invalid interlink category mappings discovered were recorded to a text file.

7. The results from the execution of the two templates (from steps 3 and 6) were compared against the invalid mapping gold standard dataset to see which of the discovered invalid mappings match the mappings in the gold standard.

**Part Two Procedure:**

The procedure for the second part of this experiment, which was concerned with the repair of invalid mappings, is as follows:

1. For each of the invalid vocabulary transformation category mappings discovered (from the first part of the experiment), a SUMMR Use Case Template 6 specialised for SCMR was prepared for its repair. In each template, the named graph of the SCMR vocabulary mapping dataset, the invalid mapping identifier, the invalid vocabulary term (to delete) and the new vocabulary term (to insert in place of the invalid vocabulary term) were specified. These were specified for the $NAMED_GRAPH, $MAP_ID, $MAPPINGPROPERTY_TO_DELETE and the $MAPPINGPROPERTY_TO_INSERT placeholders in the template. Each of the templates were then executed.

2. The instance data related to the vocabulary transformation category mappings (collected during the creation of the mapping execution results gold standard dataset) was loaded into a fifth named graph in the triple-store.

3. Each of the repaired vocabulary transformation category mappings (Part two procedure, Step 1) were executed against the instance data and the execution results were recorded to a text file.

4. For each of the invalid interlink category mappings discovered (from the first part of the experiment), a SUMMR Use Case Template 6 specialised for SCMR was prepared for its repair. In each template, the named graph of the SCMR interlink dataset, the invalid mapping identifier, the invalid instance URI (to delete) and the new URI (to insert in place of the invalid URI) were specified. These were specified for the $NAMED_GRAPH, $MAP_ID, $MAPPINGPROPERTY_TO_DELETE and the $MAPPINGPROPERTY_TO_INSERT placeholders in the template. Each of these templates were executed.
5. Each of the repaired interlink category mappings were executed and the execution results - which is a triple, were recorded to a text file.

6. The execution results from the repaired vocabulary transformation category mappings and the repaired interlink category mappings were compared to the mapping execution results gold standard dataset, triple for triple to ensure the results match.

6.4.6. Results

This sub-section presents the results obtained by performing this experiment. Recall and precision are used as metrics to judge the ability of SUMMR Use Case Template 5 specialised for SCMR to support discovery of invalid mappings, where:

- **Recall**: The ability to discover and retrieve correctly invalid mappings.
- **Precision**: The ability to withhold still valid mappings.

The results of comparing the execution results from SUMMR Use Case Template 5 specialised for SCMR to the invalid mapping gold standard dataset are displayed in Table 6.10 below:

<table>
<thead>
<tr>
<th>Invalid Mapping ID Number</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vocabulary Mapping</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3, 26, 55, 56, 59, 60, 61, 62, 67, 68</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Interlink</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9, 11, 16, 21, 26, 28, 33</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

To test the ability of SUMMR Use Case Template 6 to support the repair of a mapping, the execution results from the repaired vocabulary transformation category mappings and the repaired interlink category mappings were compared against the mapping execution results gold standard dataset to check that the results were identical. The comparison results for the repaired vocabulary transformation category mappings are presented in Table 6.11 below:

<table>
<thead>
<tr>
<th>Repaired Vocabulary Mapping Comparison Result</th>
<th>Repaired Vocabulary Mapping ID Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identical</td>
<td>3, 26, 55, 56, 59, 60, 61, 62, 67, 68</td>
</tr>
<tr>
<td>Not Identical</td>
<td>none</td>
</tr>
</tbody>
</table>
The comparison results for the repaired interlink category mappings are presented in Table 6-12 below:

<table>
<thead>
<tr>
<th>Repaired Interlinks Comparison Result</th>
<th>Repaired Interlink ID Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identical</td>
<td>9, 11, 21, 26, 33</td>
</tr>
<tr>
<td>Not Identical</td>
<td>16, 28</td>
</tr>
</tbody>
</table>

6.4.7. Discussion

As can be seen from the results in Table 6-10, SUMMR Use Case Template 5 specialised for SCMR supported the discovery of all possible invalid vocabulary transformation category mappings and invalid interlink category mappings with no false positives. The results in Table 6-11 show that SUMMR Use Case Template 6 specialised for SCMR was able to support the repair of all of the invalid vocabulary transformation category mappings. The results in Table 6-12 show that SUMMR Use Case Template 6 specialised for SCMR was able to support the repair of five of the invalid interlink category mappings, while two - number 16 and 28 could not be repaired. Note that interlink number 16 was deemed an error in the interlink category mapping dataset as it states that “Google” (which is not a pharmaceutical company) from the DBpedia dataset is the same as (owl:sameAs) <http://wifo5-04.informatik.uni-mannheim.de/dailymed/resource/organization/> from the DailyMed dataset, which does not specify an organisation in the URI. While it was possible to detect this interlink category mapping as invalid, it was not repaired due to it being an error. For interlink number 28 no new instance URI could be found that could be used to repair it.

These results indicate that for the datasets tested, SUMMR Use Case Template 5 specialised for SCMR can effectively be used to support the detection of invalid vocabulary transformation category mappings and interlink category mappings that are represented in SCMR. The results also show that for the datasets tested, SUMMR Use Case Template 6 specialised for SCMR can effectively be used to support the repair of invalid vocabulary transformation and interlink category mappings (when a new vocabulary term or instance URI has been found to replace the invalid ones) that are represented in SCMR.

It is important to note that in the SCMR vocabulary mapping dataset and the SCMR interlink dataset, no cases were found where a vocabulary term or a dataset instance changed semantically and where the URI did not change or a term or instance changed
location or was deleted and the old URI still exists (this limitation of SUMMR was discussed in Section 5.6). If either of these cases did occur in either of these two datasets, SUMMR Use Case Template 5 would likely not be able to correctly detect these invalid mappings caused by these changes.

This experiment has provided evidence that SUMMR mapping maintenance templates - based on standard SPARQL queries can be used to support the detection and repair of invalid interlink category mappings that are represented in SCMR. This provided confidence that the templates could work effectively when applied to larger scale problems - such as validating mappings that number into the millions that are likely to be encountered between some Linked Data datasets. The case study evaluation (described after the next sub-section) describes how SUMMR Use Case Template 5, via the SUMMR Interlink Validation Tool (described in Section 5.5.1), was applied for this purpose in the DBpedia interlink management process.

6.4.8. Conclusion

This section has presented the third experiment of this thesis which evaluated the ability of SUMMR query templates to support mapping maintenance and reuse tasks 4 and 5. The evidence gathered from performing the experiment indicate that the first hypothesis H5 is true, as SUMMR Use Case Template 5 specialised for SCMR achieved a recall and precision of 1.0 when discovering invalid vocabulary transformation and interlink category mappings. These results indicate that SUMMR Use Case template 5 specialised for SCMR can effectively be used to support the detection of invalid mappings when a vocabulary transformation category term (for vocabulary transformation category mappings) or a dataset instance (for interlink category mappings) has changed syntactically. Evidence gathered also indicates the second hypothesis H6 is also true, as SUMMR Use Case 6 specialised for SCMR was able to support the repair all of the invalid mappings found (from the first part of the experiment) that were possible for repair. This indicates that SUMMR Use Case Template 6 specialised for SCMR can effectively support the alteration of a vocabulary transformation category mapping or interlink category mapping represented in SCMR for the purposes of repair.

Overall this experiment has provided more evidence about the effectiveness of SUMMR templates. Through the use of SUMMR Use Case Templates 5 and 6, it has been shown that mapping maintenance and reuse tasks 4 and 5 can be performed. In
combination with the second experiment, evidence suggests that SUMMR templates are capable of supporting all five of the of the mapping maintenance and reuse tasks.

6.5. Case Study: Applying SUMMR in the DBpedia Interlink Management Process

This section presents a case study that shows the usefulness that has been observed in applying the SUMMR methodology to the DBpedia dataset release process for interlink category mapping validation. An introduction to the case study is presented in Section 6.5.1. The interlink management process of the DBpedia dataset is presented in Section 6.5.2. Where SUMMR fits in the DBpedia interlink management process is presented in Section 6.5.3. The approach and procedure undertaken is described in Section 6.5.4. Results and analysis are presented in Section 6.5.5 and the section finishes with a conclusion in Section 6.5.6.

6.5.1. Introduction

The previous three evaluations have tested different aspects of the SUMMR methodology in lab-based experiments. In this case study, a part of the SUMMR approach has been applied to a real world situation - management of a Linked Data dataset, in particular to show its usefulness for supporting the detecting of invalid interlink category mappings. The dataset that SUMMR was applied to is the DBpedia dataset.

DBpedia is a very large (with triple numbers ranging in the billions) Linked Data dataset that is extracted from the structured information on the Wikipedia website. The DBpedia community create uplift mappings, which are used to transform the knowledge from Wikipedia articles to RDF - described according to the DBpedia vocabulary. While Wikipedia articles mostly consist of unstructured information - in the form of free text, some articles can contain structured information in the form of infoboxes, article category, related images, links to external webpages and article references. It is this structured information that the DBpedia community exploit with the uplift mappings (for clarity, SUMMR is not concerned with validating the uplift mappings, it is concerned with the outgoing interlink category mappings that the DBpedia dataset maintains - as discussed below).

The DBpedia dataset has become one of the biggest hubs in the Linked Open Data Cloud and hence, many other datasets interlink to DBpedia. These external (non-
DBpedia) datasets create interlink category mappings from similar instances in their dataset to instances in the DBpedia dataset. In turn, DBpedia also provides outgoing interlink category mappings to multiple external datasets. For example, the DBpedia v.2015-04 release contained over 27 million interlink category mappings to 36 external datasets. DBpedia interlink category mapping quality is based on several factors. One of the issues that can arise with the DBpedia outgoing interlinks is they can become invalid if an instance URI that an interlink category mapping contains changes, caused by changes in the datasets the instance URIs reference. Improving the quality (correctness) of these mappings through detecting invalid interlink category mappings using the SUMMR methodology is the focus of this case study.

Currently, the DBpedia dataset is released on a biannual basis. For each release external interlink category mappings are freshly generated to the GeoNames, FreeBase and YAGO (and sometime Flickr Wrappr) datasets, the remainder are not freshly generated. The non-freshly generated interlink category mappings to external datasets are contributed by the wider DBpedia community. The community can submit sets of interlink category mappings (DBpedia as the source and another dataset as the target, stored in an RDF serialisation file such as ‘.nt’ or ‘.ttl’) to an interlink repository - where they are then added to a subsequent DBpedia release. If no new set of interlink category mappings have been submitted to the interlink repository, then interlink category mappings are copied over from the previous DBpedia release. Whether the non-freshly generated interlink category mappings come from the interlink repository or copied over from the previous DBpedia release, no measures are taken to ensure that they are valid. Validity should be checked in relation to the DBpedia dataset and the external dataset the interlink category mappings reference (when possible to access the external dataset).

In this case study, the SUMMR Use Case Template 5 was applied to the DBpedia dataset release process for the purpose of interlink category mapping validation. This was applied through the SUMMR Interlink Validation Tool. The SUMMR Interlink Validation Tool was developed as part of this case study. Through consultation with the DBpedia dataset maintainers at the University of Leipzig in Germany, requirements for the tool were gathered and an understanding of the DBpedia dataset release process was obtained. This then led to determining where the tool would fit in the release process (described below). The tool (see Section 5.5.1) was then developed by the author of this thesis. The tool was used to detect interlink category mappings that are classed as
invalid and still valid in relation to the DBpedia v.2015-10 release. The tool is still being used for this and other purposes.

6.5.2. DBpedia Interlink Management Process

This sub-section provides an overview of how interlink category mappings are generated and maintained for each release of the DBpedia dataset. The current DBpedia dataset publication cycle consists of two phases: the extraction phase and the post-processing phase. The extraction phase is concerned with the extraction of information from Wikipedia and representing it in RDF and is not discussed further here. The post-processing phase is concerned with tasks such as quality checking and the management of interlink category mappings. This second phase is focused upon in this case study as it includes interlink category mapping management.

There are three ways that interlink category mappings are supplied for a new DBpedia release: script-based generation, user submission or reuse. For a new release of DBpedia, interlink category mappings are freshly created to three (sometime four) external datasets via custom scripts\(^{15}\). These three external datasets are Freebase, GeoNames and YAGO (and sometimes Flickr Wrappr). The remaining interlink category mappings to be used are supplied by external contributors who submit interlink category mappings to a repository on GitHub\(^{16}\) or they are copied over from the previous release. If a set of interlink category mappings to an external dataset do not exist in the interlink repository but exist in the previous release of DBpedia, then these interlink category mappings will be copied over to be used in the latest release. Figure 6-2 displays the interlink management process in the DBpedia dataset release cycle.

![Figure 6-2. DBpedia Interlink Management Process.](image)

\(^{15}\) These are bash shell scripts, developed and maintained by the maintainers of the DBpedia dataset.

\(^{16}\) [https://github.com/dbpedia/dbpedia-links](https://github.com/dbpedia/dbpedia-links)
As part of the requirements analysis of this case study, a problem was identified with the interlink management process: the interlink category mappings that come from the interlink repository or the interlink category mappings that are copied over from the previous release are not checked to ensure they are still valid in relation to the new release of DBpedia. This can result in invalid interlink category mappings being published as part of the dataset release. For the purpose of validating interlink category mappings, that are to be published as part of the dataset, the SUMMR Interlink Validation Tool was applied to the interlink management process.

6.5.3. SUMMR Applied to the Process

This sub-section describes where the SUMMR Interlink Validation Tool fits into the DBpedia interlink management process for the purpose of validating interlink category mappings and it is depicted in Figure 6-3 below:

![Figure 6-3. DBpedia Interlink Management Process with Interlink Validation Provided by the SUMMR Interlink Validation Tool.](image)

The SUMMR Interlink Validation Tool (described in detail in Section 5.5.1) is a command line, java-based tool. Through consultation with the DBpedia team at the University of Leipzig, it was determined that the tool would be used to validate the interlink category mappings that come from the interlink repository and the interlink category mappings that are be copied over from the previous DBpedia release. Since the interlink category mappings between DBpedia and the FreeBase, GeoNames and YAGO (and sometimes Flickr Wrappr) datasets are freshly generated, they do not need to be validated as they have been created for the newest versions of these datasets. The interlink category mappings from the interlink repository and the interlink category mappings copied over from the previous release do need to be validated as they would
have been created before the newest DBpedia release so they could be invalid in relation to that latest release.

The interlink category mappings are validated through SUMMR use case template 5. This process involves checking that the DBpedia instance URIs of the interlink category mappings exist in the new release of the DBpedia dataset. The external dataset instance URIs of the interlink category mappings can also be checked to ensure they exist in their respective external dataset. If a DBpedia instance URI from an interlink category mapping does not exist in the new DBpedia release, and (or) an external instance URI does not exist in its respective dataset, then that interlink category mapping is classed as invalid.

6.5.4. Approach and Procedure

This sub-section details the approach taken to validate the interlink category mappings in the DBpedia v.2015-10 release with the SUMMR Interlink Validation Tool.

For the DBpedia v.2015-10 release, the interlink category mapping sets (interlinks between DBpedia as the source and another dataset as the target) from the previous DBpedia release (v.2015-04) were copied to the new release except for the interlink category mappings to FreeBase, GeoNames, YAGO as they were freshly generated. The interlink category mappings to Flickr Wrappr were also freshly generated for the v.2015-10 release. The SUMMR Interlink Validation Tool was used to detect all invalid interlink category mappings from the interlink sets. It checked if the DBpedia instance URIs in the interlink sets exist in the DBpedia v.2015-10 release. Interlink sets were also checked against their respective external datasets - when it was possible to access that external dataset via a Federated SPARQL query.

A potential source of interlink category mapping error was brought to the attention of the author of thesis by the maintainers of the DBpedia dataset. For the DBpedia v.2015-10 release, a transition was taken from representing instances in the dataset with Uniform Resource Identifiers (URI) to representing them with Internationalized Resource Identifiers (IRI). URIs and IRIs that are not identical as the strings are not the same, even when they are equivalent. For example, dbr:André_Gide is the IRI form of the dbr:Andr%C3%A9_Gide URI but links to the latter are not valid links to the former. This change had the potential to impact existing interlink category mappings. To help alleviate problems that could arise from the switch from URIs to IRIs, the DBpedia dataset maintainers created and published a set of URI-to-IRI mappings with the
v.2015-10 release. In this case study, influenced by this change, the interlink sets were validated twice: (i) without considering the URI-to-IRI mappings and (ii) taking the URI-to-IRI mappings into consideration. By doing this, the impact of the transition from URIs to IRIs had on the interlink category mappings could be measured.

The following outlines the procedure that was undertaken in this case study. The list of external datasets that DBpedia provides interlink category mappings to can be found under the “Links to other datasets” section v.2015-10 DBpedia release download page. Out of the 36 total external datasets, 32 datasets consisting of 1,673,634 interlink category mappings were validated with the SUMMR Interlink Validation Tool. The four interlink sets that were not validated were the sets that were freshly generated for the v.2015-10 DBpedia release (interlinks from DBpedia to the FreeBase, GeoNames, YAGO and Flickr Wrappr).

1. The DBpedia v.2015-10 release was loaded into a Virtuoso triple-store (Excluding the URI-to-IRI mappings).
2. The 32 interlink sets to be validated were downloaded from the DBpedia download page (see Footnote 19).
3. The SUMMR Interlink Validation Tool was executed to validate the 32 interlink sets. All of the interlink sets were checked against the DBpedia v.2015-10 release. Seven of the interlink sets were additionally checked against their respective external dataset as these were they only datasets were accessible through a Federated SPARQL query was possible. See Table 6-13 for a breakdown of which interlinks sets were and were not checked against their external datasets.
4. The results from the execution of the tool were recorded.
5. The URI-to-IRI mappings were loaded into the triple-store.
6. Repeat of steps 3-4 to validate the interlink sets where the URI-to-IRI mappings are considered.

6.5.5. Results and Analysis

This sub-section presents the results obtained from performing this case study and an analysis about those results. There are 27,260,134 interlink category mappings to 36 external datasets published in the v.2015-04 DBpedia release. The majority of these interlink category mappings (25,586,500) are to the FreeBase, GeoNames and YAGO

---

(which are freshly generated for each release of the DBpedia dataset) and Flickr Wrappr datasets. In this case study, the interlink category mappings to the remaining 32 external data sets were validated which equates to 1,673,634 or 6.14% of the total mappings validated. See Table 6-13 for a detailed breakdown of the results.

Table 6-13. Detailed breakdown of the interlink sets validated by SUMMR QT5, categorized by external dataset name and indicating the number of interlinks per set, whether the interlink set was validated against an external dataset and the number and percentage of invalid interlink category mappings discovered per set, when the URI-to-IRI mapping were not considered and when they were considered.

<table>
<thead>
<tr>
<th>External Dataset</th>
<th>Number of Interlinks</th>
<th>Externally Validated</th>
<th>Number and Percentage of Invalid Interlinks (No URI-to-IRI Mappings)</th>
<th>Number and Percentage of Invalid Interlinks (With URI-to-IRI Mappings)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amsterdam Mus</td>
<td>627</td>
<td>No</td>
<td>74 (11.8%)</td>
<td>10 (1.6%)</td>
</tr>
<tr>
<td>BBCWildlife</td>
<td>444</td>
<td>No</td>
<td>4 (0.9%)</td>
<td>1 (0.2%)</td>
</tr>
<tr>
<td>Bookmashup</td>
<td>8,903</td>
<td>No</td>
<td>244 (2.74%)</td>
<td>168 (1.88%)</td>
</tr>
<tr>
<td>Bricklink</td>
<td>10,090</td>
<td>No</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>CIAFactbook</td>
<td>545</td>
<td>No</td>
<td>7 (1.28%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>CORDIS</td>
<td>314</td>
<td>Yes</td>
<td>24 (7.64%)</td>
<td>19 (6.1%)</td>
</tr>
<tr>
<td>DailyMed</td>
<td>894</td>
<td>Yes</td>
<td>894 (100%)</td>
<td>894 (100%)</td>
</tr>
<tr>
<td>DBLP</td>
<td>196</td>
<td>No</td>
<td>5 (2.5%)</td>
<td>4 (2%)</td>
</tr>
<tr>
<td>DBTune</td>
<td>838</td>
<td>No</td>
<td>60 (7.1%)</td>
<td>33 (4%)</td>
</tr>
<tr>
<td>Diseasome</td>
<td>2,301</td>
<td>No</td>
<td>90 (3.9%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>DrugBank</td>
<td>4,845</td>
<td>Yes</td>
<td>4,845 (100%)</td>
<td>4,845 (100%)</td>
</tr>
<tr>
<td>EUNIS</td>
<td>11,235</td>
<td>Yes</td>
<td>219 (1.9%)</td>
<td>206 (1.8%)</td>
</tr>
<tr>
<td>EuroStatLinkedSt</td>
<td>253</td>
<td>Yes</td>
<td>2 (0.8%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>EuroStatWBSG</td>
<td>137</td>
<td>No</td>
<td>2 (1.4%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>GADM</td>
<td>42,332</td>
<td>No</td>
<td>8 (0.02%)</td>
<td>8 (0.02%)</td>
</tr>
<tr>
<td>GeoSpecies</td>
<td>15,974</td>
<td>No</td>
<td>109 (0.7%)</td>
<td>2 (0.01%)</td>
</tr>
<tr>
<td>GHO</td>
<td>196</td>
<td>No</td>
<td>5 (2.5%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Gutenberg</td>
<td>2,510</td>
<td>No</td>
<td>163 (6.5%)</td>
<td>5 (0.2%)</td>
</tr>
<tr>
<td>ItalianPublicSchls</td>
<td>5,822</td>
<td>No</td>
<td>113 (1.9%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>LinkedGeoData</td>
<td>103,633</td>
<td>Yes</td>
<td>27,194 (26.2%)</td>
<td>4,559 (16.9%)</td>
</tr>
<tr>
<td>LMDB</td>
<td>13,758</td>
<td>No</td>
<td>372 (2.7%)</td>
<td>8 (0.06%)</td>
</tr>
<tr>
<td>MusicBrainz</td>
<td>22,981</td>
<td>No</td>
<td>1,006 (4.4%)</td>
<td>524 (2.3%)</td>
</tr>
<tr>
<td>NewYorkTimes</td>
<td>9,678</td>
<td>No</td>
<td>35 (0.4%)</td>
<td>6 (0.06%)</td>
</tr>
<tr>
<td>OpenCyc</td>
<td>27,104</td>
<td>No</td>
<td>780 (2.9%)</td>
<td>167 (0.6%)</td>
</tr>
<tr>
<td>OpenEI</td>
<td>678</td>
<td>Yes</td>
<td>20 (2.9%)</td>
<td>4 (0.6%)</td>
</tr>
<tr>
<td>Revyu</td>
<td>6</td>
<td>No</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>SIDER</td>
<td>1,969</td>
<td>No</td>
<td>11 (0.5%)</td>
<td>2 (0.1%)</td>
</tr>
<tr>
<td>TCMGene</td>
<td>904</td>
<td>No</td>
<td>2 (0.2%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>UMBEL</td>
<td>896,423</td>
<td>No</td>
<td>98,908 (11%)</td>
<td>34,339 (3.8%)</td>
</tr>
<tr>
<td>USCensus</td>
<td>12,592</td>
<td>No</td>
<td>2 (0.01%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>WikCompany</td>
<td>8,348</td>
<td>No</td>
<td>384 (4.6%)</td>
<td>355 (4.2%)</td>
</tr>
<tr>
<td>WordNet</td>
<td>467,101</td>
<td>No</td>
<td>33,809 (7.2%)</td>
<td>7,265 (1.5%)</td>
</tr>
<tr>
<td><strong>TOTAL:</strong></td>
<td><strong>1,673,634</strong></td>
<td></td>
<td><strong>169,391</strong></td>
<td><strong>53,418</strong></td>
</tr>
<tr>
<td><strong>PERCENTAGE:</strong></td>
<td><strong>10.12%</strong></td>
<td></td>
<td><strong>3.19%</strong></td>
<td></td>
</tr>
</tbody>
</table>
10.12% of the interlink category mappings checked were classed as invalid when the URI-to-IRI mappings were not considered compared to 3.19% when they were considered. This difference of 6.93% shows the importance of the URI-to-IRI mappings in relation to the interlink sets for the DBpedia v.2015-10 release. Ultimately, 3.19% of the interlink category mappings were found to be invalid and these invalid interlink category mappings were removed from the published v.2015-10 release.

Table 6.14 presents a breakdown for the interlink sets that were externally checked, how many of the invalid interlink category mappings found were caused due to changes in the DBpedia dataset or the external dataset. As can be seen a total of 10,239 interlink category mappings were invalidated by changes in external datasets. When considering that 53,418 invalid interlink category mappings were found in total, 10,239 make up 19.17% of that total. This shows the importance of checking both datasets when validating interlink category mappings. However, this is not always possible as up-to-date dumps of external datasets are not always available or an external dataset is not behind a SPARQL endpoint with federated query access enabled. Therefore it is unknown how many of the non-invalid interlink category mappings found that were not externally checked, would be found to be invalid if they were externally checked.

<table>
<thead>
<tr>
<th>External Validated Dataset</th>
<th>Number of Interlinks Invalidated by changes in DBpedia (No URI-to-IRI Mappings)</th>
<th>Number of Interlinks Invalidated by changes in DBpedia (With URI-to-IRI Mappings)</th>
<th>Number of Interlinks Invalidated by changes in External Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORDIS</td>
<td>24</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>DailyMed</td>
<td>0</td>
<td>0</td>
<td>894</td>
</tr>
<tr>
<td>DrugBank</td>
<td>23</td>
<td>23</td>
<td>4,822</td>
</tr>
<tr>
<td>EUNIS</td>
<td>217</td>
<td>204</td>
<td>2</td>
</tr>
<tr>
<td>EuroStatLinkedSt</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LinkedGeoData</td>
<td>22,671</td>
<td>36</td>
<td>4,523</td>
</tr>
<tr>
<td>OpenEI</td>
<td>20</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td><strong>TOTAL:</strong></td>
<td><strong>22,957</strong></td>
<td><strong>276</strong></td>
<td><strong>10,239</strong></td>
</tr>
</tbody>
</table>

6.5.6. Conclusion

This section has presented a case study that involved applying the SUMMR methodology, via the SUMMR Interlink Validation Tool, to the interlink management process of the DBpedia dataset. This was done to show the usefulness of SUMMR in a
real Linked Data dataset management situation. SUMMR supported the validated 1,673,634 interlink category mappings that were to be published as part of the DBpedia v.2015-10 release. Of these 1,673,634 interlink category mappings, 53,418 (3.19%) were found to be invalid resulting in SUMMR having a direct impact on the quality of the DBpedia v.2015-10 release. The results from this case were published in the first paper described in Section 1.2.3.

The case study has highlighted the importance of interlink category mapping validation for Linked Data datasets and the usefulness of approaches, like SUMMR, to support that detection. The development of the SUMMR Interlink Validation Tool was influenced by this case study. It is an open-source tool that was designed to operate with any Linked Data dataset (not just DBpedia) for interlink category mapping validation and it is hoped it will be adopted by other dataset maintainers to support them with interlink category mapping validation. The Tool is now used by the DBpedia project during the release process of a new version of the DBpedia dataset.

Please note that the author of this thesis was credited for the invalid interlink category mapping detection (as described in this case study) by the DBpedia team, in an announcement email, to the DBpedia community, for the release of the v.2015-10 DBpedia dataset. A copy of this email can be found in Appendix C of this thesis.

6.6. Chapter Summary

In summary, this chapter has presented four evaluations of the SUMMR methodology. The first evaluation was an experiment with the objective of testing the expressivity of the SCMR to represent executable vocabulary transformation category mappings and encode those mappings in RDF. The experiment results have shown that for the vocabulary transformation category mapping dataset used in the experiment that: (i) the SCMR is equally as expressive as the R2R Mapping Language for representing executable vocabulary transformation category mappings as it was able to represent 100% of the mappings from the dataset and (ii) the SCMR can encode 100% those mappings in RDF.

The objective of the second experiment was to test the mapping maintenance and reuse affordance of the SCMR and the ability of the SUMMR methodology query templates to support a potential user to perform mapping maintenance and reuse task numbers 1 to 3. The experiment results show that compared to the R2R Mapping Language, the SCMR affords a greater number of mapping maintenance and reuse tasks
to be performed using SUMMR templates. The results also show that SUMMR templates specialised for SCMR can effectively perform mapping maintenance and reuse task numbers 1 to 3 through being able to perform the mapping retrieval use cases in the experiment.

The third evaluation was an experiment with the objective of testing the ability of the SUMMR query templates to support a potential user to perform mapping maintenance and reuse task numbers 4 and 5. The results show that SUMMR Use Case Template 5 specialised for SCMR can effectively perform mapping maintenance and reuse task number 4 through correctly discovering 100% of the invalid vocabulary transformation category mappings and interlink category mappings from the datasets used in the experiment. The results also show that SUMMR Use Case Template 6 specialised for SCMR can effectively perform mapping maintenance and reuse task number 5 through correctly repairing (when possible) all of the invalid vocabulary transformation category mappings and interlink category mappings discovered in the experiment.

Finally, the case study which involved applying the SUMMR methodology in the DBpedia project interlink management process. This case study has shown the usefulness of SUMMR in this scenario, where it discovered 53,418 invalid interlink category mappings, from the interlink category mappings that were to be published as part of the DBpedia v.2015-10 release.

Overall the evaluations has shown that the SUMMR methodology with its templates and mapping representation is effective at performing maintenance and reuse over Linked Data mappings. It has also been shown, through the case study that is applicable to and useful in a real world situation – related to data quality of Linked Data datasets.
7. Conclusion

This chapter draws the final conclusion from the research presented in this thesis. Section 7.1 discusses to what extent the research objectives of this thesis (as set out in Chapter 1) have been achieved. In Section 7.2 the contributions of the research are revisited. Section 7.3 presents topics for possible further work based on the research described here. To end, final remarks are presented in Section 7.4.

7.1. Evaluation of Achievement of Research Objectives

Five research objectives were posed, in Section 1.2.2, to address the research question investigated in this thesis. The sub-sections below sequentially present an analysis of the extent to which each research objective was achieved.

7.1.1. Objective 1: Review Existing Approaches for Mapping Maintenance and Reuse

The first research objective (RO1) was to perform a state of the art review of existing approaches for mapping maintenance or mapping reuse, or both, and existing approaches for representing vocabulary transformation category mappings and RDF interlink category mappings.

This research objective was achieved through the review of the state of the art described in Chapter 3 in this thesis. This review examined 13 existing approaches that support mapping maintenance and reuse. These approaches were examined to determine the mapping maintenance or reuse use cases that they supported. The review also examined eight existing approaches for representing Linked Data vocabulary transformation and interlink category mappings. These mapping representations were reviewed on their characteristics such as: what they are designed to represent, if they support annotation with meta-data, how they are encoded and if they are executable.

The state of the art review identified a total of six mapping maintenance and reuse use cases that were supported by the existing approaches. These six use cases were then broken down into individual tasks that are needed to perform the use cases. A total of five mapping maintenance and reuse tasks were derived. From the review, a gap in the state of the art was identified in that none of the existing approaches support all six of the mapping maintenance and reuse use cases and hence - the five tasks. The review of existing approaches for vocabulary transformation and interlink category mapping
representation has provided information on the characteristics of these mapping representations. This information was considered for evaluating which representation to use in the design of a new approach for mapping maintenance and reuse, and specifically the decision whether to choose an existing mapping representation or design a new one.

7.1.2. Objective 2: Design a Query-based Methodology for Mapping Maintenance and Reuse

The second research objective (RO2) was to design a query-based methodology for undertaking both mapping maintenance and reuse. This will include an appropriate mapping representation and query templates to perform maintenance and reuse.

This research objective was achieved through the creation of the SPARQL Usage for Mapping Maintenance and Reuse (SUMMR) methodology and the SPARQL Centric Mapping Representation (SCMR), which are described in Chapter 5. The aim of SUMMR is to provide a methodology to support users in performing Linked Data mapping maintenance and reuse. SUMMR provides two sets of SPARQL query templates that are to be performed over an RDF-based mapping representation. The first set of query templates, the SUMMR Task Templates, consists of five templates which are for performing mapping maintenance and reuse tasks (derived from the state of the art review). The second set of query templates, the SUMMR Use Case Templates, consists of six templates for performing mapping maintenance and reuse use cases (identified from the state of the art review). Recall from Chapter 5 (and as illustrated in Figure 5-2) that task templates are combined to make the use case templates. The SUMMR templates can be specialised to work with different RDF-based mapping representations. However, SUMMR does provide a mapping representation - the SCMR. The SCMR is an RDF-based mapping representation that can represent both vocabulary transformation and interlink category mappings. It represents mapping properties at a fine grained level and supports annotation with mapping meta-data. The fine granularity affords greater analysis capabilities upon the SCMR by SPARQL queries (and hence, SUMMR templates), as evidenced in the second experiment (Section 6.3).
7.1.3. Objective 3: Implement the Query-based Methodology

The purpose of the third research objective (RO3) was to design a prototype software tool that implements the new query-based methodology from RO2.

In Section 5.4.2, multiple possible implementations of the SUMMR methodology were proposed, ranging from tools that facilitated the creation of SCMRs to tools which fully implement all of SUMMR’s functionality with sophisticated user interfaces. This research objective has been partially achieved through the development of the SUMMR Interlink Validation Tool, which is described in Section 5.4.1. The SUMMR Interlink Validation Tool is a prototype, open-source tool that implements SUMMR Use Case Template 5 - specifically for detecting invalid interlink category mappings. The purpose of the tool is to validate sets of interlink category mappings between a source Linked Data dataset and multiple target Linked Data datasets. The tool was developed to test (a part of) the SUMMR methodologies usefulness in a real Linked Data dataset management situation. The tool was used to perform interlink category mapping validation in the DBpedia dataset, which was not previously performed.

7.1.4. Objective 4: Evaluate the Ability of the Query-based methodology to Support Performing Mapping Maintenance and Reuse

The fourth research objective (RO4) was stated as follows: evaluate the query-based methodology from RO2 in lab-based experiments. This will involve evaluating (i) the expressivity of the mapping representation used by the methodology, and (ii) the ability of the query templates to support to support mapping maintenance and reuse tasks.

This research objective was achieved through performing three lab-based experiments.

Experiment 1:
The first experiment, described in Section 6.2, was concerned with testing the expressivity of the SPARQL Centric Mapping Representation (SCMR) in two parts. The first part was concerned with testing SPARQL queries (which the SCMR uses) to express executable vocabulary transformation category mappings. This was tested by comparing SPARQL CONSTRUCT queries against an existing mapping representation (for representing vocabulary transformation category mappings) - the R2R Mapping Language. It was found that the SPARQL CONSTRUCT were equally as expressive at representing the mappings that the R2R Mapping Language can.
The second part was concerned with testing the SPIN SPARQL syntax (which the SCMR uses) to encoding SPARQL-based vocabulary transformation category mappings in RDF. This was done by testing the expressivity of the SPIN SPARQL syntax with regard to encoding SPARQL CONSTRUCT queries in RDF. It was found that the SPIN SPARQL syntax was sufficiently expressive to represent the queries in RDF. This is important as it the fine granularity of the mapping representation affords more mapping maintenance and reuse tasks to be performed over it using SUMMR templates.

**Experiment 2:**
The second lab-based experiment, described in Section 6.3, tested two aspects of SUMMR. First was the affordance offered by the SCMR for performing mapping maintenance and reuse tasks. Second was the ability of SUMMR templates to support a potential user to perform mapping reuse tasks (tasks 1 to 3) of the five mapping maintenance and reuse tasks derived from the state of the art review. This was tested using 12 mapping retrieval use cases (which encompassed tasks 1 to 3), which were originally motivated in the research of Thomas et al. [Th11]. Two sets of SUMMR Task Templates 1 to 3, specialised for both SCMR and the R2R Mapping Language, were used to perform the twelve mapping retrieval use cases. From this experiment it was found that SUMMR Task Template 2 could not be performed over mappings represented in the R2R Mapping Language but could be performed over mappings represented in SCMR. This was found to be caused by the granularity difference between how the two mapping representations are modelled. The R2R Mapping Language is more coarse grain compared to the SCMR.

**Experiment 3:**
The third lab-based experiment, described in Section 6.4, tested the ability of SUMMR templates to support a potential user to perform the remaining two mapping maintenance and reuse tasks - tasks 4 and 5. This was tested in two parts.

In part 1, SUMMR Use Case Template 5 (see Table 7-1 below) specialised for SCMR was used. It was used to discover both invalid vocabulary transformation and invalid interlink category mappings, both represented in SCMR. It was found that SUMMR Use Case Template 5 successfully discovered all vocabulary transformation and interlink category mapping from the datasets used in the experiment.
### Table 7-1. SUMMR Use Case Templates used in parts 1 and 2 of experiment 3.

<table>
<thead>
<tr>
<th>Experiment 3 Part Number</th>
<th>SUMMR Use Case Template Used</th>
<th>Use Case Name / Template Purpose</th>
<th>Maintenance and Reuse Task Involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1</td>
<td>Use Case Template 5</td>
<td>Discover invalid mappings due to changes in datasets a mapping references</td>
<td>1, 2, 4</td>
</tr>
<tr>
<td>Part 2</td>
<td>Use Case Template 6</td>
<td>Repair of an invalid mapping</td>
<td>5</td>
</tr>
</tbody>
</table>

In part 2, SUMMR Use Case Template 6 (see Table 7-1) specialised for SCMR was used. It was used to repair the invalid mappings found from the first part of the experiment and it was found that SUMMR Use Case Template 6 could successfully repair all the invalid mappings (when it was possible for repair).

#### 7.1.5. Objective 5: Evaluate the Usefulness of the Query-based Methodology in a Real World Linked Data Situation

Research objective five (RO5) was to evaluate the utility of the query-based methodology from RO2 in a case study, where the software tool from RO3 will be deployed in the release process of a real world Linked Data dataset.

This research objective was achieved through performing the case study, described in Section 6.5. The case study involved applying the SUMMR methodology, through the SUMMR Interlink Validation Tool, to the live DBpedia dataset release cycle for the purpose of validating interlink category mappings that are to be used in a new release of the DBpedia dataset. The tool was used to validate 1,679,634 interlink category mappings that were to be published in the v.2015-10 DBpedia dataset release. These interlink category mappings were between DBpedia and 32 external Linked Data datasets. The SUMMR Interlink Validation Tool discovered that 53,418 (3.19% of total interlink category mappings checked) of the interlink category mappings checked were classed as invalid. The SUMMR methodology (through the SUMMR Interlink Validation Tool) had a direct impact on the overall quality of the DBpedia v.2015-10 release through discovering the invalid interlink category mappings and has shown its usefulness in doing so.

#### 7.2. Contributions

This section briefly revisits the contributions from the research of this thesis, which were initially presented in Section 1.2.3.
The research of this thesis has yielded three contributions: one major contribution and two minor contributions. The major contribution is the SPARQL Usage for Mapping Maintenance and Reuse (SUMMR) methodology. SUMMR is a methodology that supports a user to perform mapping maintenance and reuse over Linked Data vocabulary transformation and interlink category mappings. SUMMR is novel in that, to the knowledge of the author, it is the only methodology that exists for Linked Data mapping maintenance and reuse. The elements of SUMMR’s design (Section 5.2) can be thought of as a set of practices that can be undertaken for mapping maintenance and reuse and SUMMR templates can be thought of as a set of procedures and rules that can be followed to actually perform it. SUMMR provides query templates to perform mapping maintenance and reuse se cases and tasks. The templates can be specialised to perform over different RDF-based mapping representations, however SUMMR itself does provide a mapping representation, which is the first minor contribution (see below). SUMMR advances the state of the art through the creation of a new taxonomy, which consists of five mapping maintenance and reuse tasks and six mapping maintenance and reuse use cases. This taxonomy can be used as a benchmark for future mapping maintenance and reuse application. The SUMMR methodology also advances the state of the art as (to the knowledge of the author) it is the only approach that supports performing all the mapping maintenance and reuse tasks and use cases derived from the state of the art.

The first minor contribution is the SPARQL Centric Mapping Representation (SCMR). SCMR is an RDF-based mapping representation designed to represent both vocabulary transformation and interlink category mappings at a fine granularity which also supports annotation with mapping meta-data. SCMR complements SUMMR-based mapping maintenance and reuse by facilitating discovery and retrieval of mappings and mapping properties due to its explicit modelling of all the mapping properties as RDF triples as opposed to modelling them as a string as is done in the R2R Mapping Language. This allows for mapping maintenance and reuse tasks to be performed over it using SUMMR templates. While SCMR was developed in conjunction with SUMMR, it is envisaged that it will be of benefit outside just mapping maintenance and reuse. Other applications which require the storage of mappings and which require fine grain the retrieval and analysis of mappings should find the SCMR of use.

The second minor contribution is the open-source SUMMR Interlink Validation Tool, which implements SUMMR Use Case Template 5 for invalid interlink category
mapping discovery. The tool was developed as part of the case study in this thesis. It was used in the DBpedia interlink management process to discover invalid interlink category mappings for the v.2015-10 DBpedia dataset release. The tool is now part of multiple technologies used by the DBpedia dataset during the process of a new release of the dataset. It has also been incorporated into the H2020 ALIGNED project’s DBpedia Phase 2 Trial Platform. The SUMMR Interlink Validation Tool is an open-source tool, which can be used to support invalid interlink category mapping detection for any Linked Data dataset that needs to maintain interlink category mappings from a source dataset to multiple target datasets.

As stated in Section 1.2.3 this research is already having impact within the research community with publications in ODBASE 2016, IEEE Semantic Computing 2015 and Extended Semantic Web Conference 2014. A journal publication targeting the Journal of Web Semantics is underway.

7.3. Further Work

This section discusses potential further work that could be undertaken for the research in this thesis.

One possible route of further work would be to develop additional implementations of SUMMR which incorporate more of its functionality. To date, the only implementation is the SUMMR Interlink Validation Tool, which currently only implements SUMMR Use Case Template 5 for discovering invalid interlink category mappings. In the Potential SUMMR Tools section (Section 5.4.2), different possibilities of such implementations were discussed, along with a single tool with the possibility of incorporating all of SUMMR’s functionality. These tools could be implemented as command line tools, or with graphical user interfaces. While beyond the scope of research of this thesis, such implementations would require user testing to evaluate their effectiveness - especially for tools with graphical user interfaces. It is likely more research would be required on how much or how little information should be displayed to a user and what is the best way to display and present that information to enable a user to make effective decisions. Similar research has been carried out on the graphical user interfaces of ontology matching tools [Fa11].

Another possible route of further work would be to complement, possible SUMMR implementations, with ontology matching tools and dataset versioning tools. As stated in Section 5.5.1, a limitation of SUMMR’s approach for detecting invalid mappings is
that it would not be able to correctly discover an invalid mapping, when a resource that the mapping references changes semantically but not syntactically. This limitation could potentially be overcome if SUMMR was used with dataset versioning tools that are capable of detecting semantic changes between different versions of dataset resources. Ontology matching tools could be used to complement SUMMR through supporting the discovery of new resources from a dataset that could be used to repair invalid mappings discovered by SUMMR. To build further on this line of future work, research into a new algorithm, which involves SUMMR, dataset versioning tools and ontology matching tools could be undertaken to develop a more automated approach for mapping maintenance, similar to DyKOSMap [Do15a] which is described in Section 3.2.1.

7.4. Final Remarks

It is hoped by the author of this thesis that SUMMR, a methodology that supports performing all the mapping maintenance and reuse use cases and tasks, discovered and derived from the state of the art review in this thesis will be of benefit to the research community. The taxonomy of tasks and use cases will be useful for other researchers who wish to design an approach for mapping maintenance and reuse as it indicates what an approach should support. Researchers could also contribute to the taxonomy with new mapping maintenance and reuse use cases they may encounter in the future - further enriching it for additional researchers. SUMMR’s functionality (as a whole or aspects of it) could also be adopted as part of larger frameworks which require the maintenance or reuse of mappings in some way.

It is also hoped that SUMMR will be of benefit to the providers and maintainers of Linked Data datasets through the adoption of SUMMR (as a whole or aspects of). SUMMR has been shown to be beneficial to the DBpedia dataset through the use of the SUMMR Interlink Validation Tool. The SUMMR Interlink Validation Tool is a free, open-source tool, which can be used for any Linked Data dataset for interlink category mapping validation. It is available for the providers and maintainers of Linked Data datasets to gain the benefit, of interlink category mapping validation that SUMMR offers.
References


[Bo10] Boran, A., O'Sullivan, D. and Wade, V. "Managing ontology based integration systems using dependencies." In *Pervasive Computing and


Schultz, A., Matteini, A., Isele, R., Bizer, C. and Becker, C. “LDIF-Linked Data Integration Framework for Large-Scale Linked Data Integration.” In


Appendix A. SUMMR Use Case Template Examples

In Chapter 5, two examples of SUMMR use case templates were provided. These examples were of SUMMR Use Case Templates 1 and 5. In this appendix, examples of the remaining four SUMMR use case templates are provided.

SUMMR Use Case Template 2:
Figure A-1 displays an example of SUMMR Use Case Template 2 specialised for SCMR. This template is concerned returning the source and target properties and any transformation properties of SCMR vocabulary transformation category mappings that adhere to certain search criteria. The criteria that needs to be specified in this example template are specified in the $SEARCH_CRITERIA variable. The $SEARCH_CRITERIA variable will be a string. The idea is that a keyword will be specified in this variable so existing mapping that contain that keyword within their source and target properties can be discovered so that the source and target properties and any transformation properties of the mappings can be returned.

01: PREFIX sp: <http://spinrdf.org/sp#>
02: PREFIX scmr: <https://www.scss.tcd.ie/~meehanal/scmr>
03: SELECT ?source_properties ?target_properties ?transformation_info
08: ?trans ?pt ?transformation_info .}
11: FILTER ( REGEX( STR(?source_properties), $SEARCH_CRITERIA ) || REGEX( STR(?target_properties), $SEARCH_CRITERIA ) )

Figure A-1. Example of SUMMR Use Case Template 2 specialised for SCMR.

SUMMR Use Case Template 3:
Figure A-2 displays an example of SUMMR Use Case Template 3 specialised for SCMR. This template is concerned with discovering mapping paths within SCMR vocabulary transformation category mappings. The template will return the source properties of one set of mappings and the target properties of another set of mappings where a mapping paths have been discovered between the two sets of mappings. The
returned source and target properties can then be used to create new mappings. The criteria that need to be specified in this example template are specified in two variables. The $SOURCE_DATASET_DOMAIN$ and the $TARGET_DATASET_DOMAIN$ variables will both be stings, indicating the domain URI of the source and target that mapping paths are to be discovered for.

```
01: PREFIX sp: <http://spinrdf.org/sp#>
02: PREFIX scmr: <https://www.scss.tcd.ie/~meehanal/scmr>
03: SELECT ?source ?target
07:   FILTER ( REGEX( STR(?source), $SOURCE_DATASET_DOMAIN ) )
13:   FILTER ( REGEX( STR(?target), $TARGET_DATASET_DOMAIN ) )
16:   FILTER(?intermediate_dataset = ?intermediate_dataset1)}
```

Figure A-2. Example of SUMMR Use Case Template 3.

SUMMR Use Case Template 4:

Figure A-3 displays an example of SUMMR Use Case Template 3 specialised for SCMR. This template is concerned with discovering back links in SCMR interlink category mappings. The template will first discover interlink category mappings between a source and target dataset, then it will return the source and target properties in reverse order so the target properties can be used as a new source and the source properties can be used as a new target for new interlink category mappings. The criteria that need to be specified in this example template are specified in two variables. The $SOURCE_DATASET_DOMAIN$ and the $TARGET_DATASET_DOMAIN$ variables will both be stings, indicating the domain URI of the source and target that interlink category mappings that are to be discovered.
SUMMR Use Case Template 6:

*Figure A-4* displays an example of SUMMR Use Case Template 6 specialised for SCMR. SUMMR Use Case Template 6 is identical to SUMMR Task Template 5 as it only contains one mapping maintenance and reuse task – task 5 which is *alteration of a mapping*. This template is concerned with changing a property in a vocabulary transformation category mapping. In this template, the criteria to specify is the mapping identifier specified as the variable $MAP_ID$. The remaining criteria to specify is the property to delete, specified as variable $MAPPING_PROPERTY_TO_DELETE$ and the property to insert, specified by the variable $MAPPING_PROPERTY_TO_INSERT$. All variables in this template will be URIs.

```
01: PREFIX sp: <http://spinrdf.org/sp#>
02: PREFIX scmr: <https://www.scss.tcd.ie/~meehanal/scmr>
03: DELETE {?map_props ?b1 $MAPPING_PROPERTY_TO_DELETE.
   ?rep scmr:sparqlQuery ?query.}
04: INSERT {?map_props ?b1 $MAPPING_PROPERTY_TO_INSERT.
   ?rep scmr:sparqlQuery ?new_query}
05: WHERE { $MAP_ID scmr:SpinRepresentation ?rep .
   BIND(( REPLACE( ?query, $MAPPING_PROPERTY_TO_DELETE,
     $MAPPING_PROPERTY_TO_INSERT )) AS ?new_query)
07: UNION {?rep sp:templates/rdf:rest*/rdf:first ?map_props.}
09: UNION {?rep sp:where/rdf:rest*/rdf:first ?map_props.}
10: ?map_props ?b1 $MAPPING_PROPERTY_TO_DELETE.}
```

*Figure A-4. Example of SUMMR Use Case Template 6.*
Appendix B. DVD Contents

The accompanying DVD with this thesis makes available all the data used in Experiments 1, 2 and 3 of this thesis.

All data used in Experiment 1, 2 and 3 of this thesis can be found in the experiment_data/experiment_1, experiment_data/experiment_2 and experiment_data/experiment_3 folders respectively. All the remaining sub-folders and files, within these three folder, have been appropriately named to make finding specific data about an experiment straight forward.
Appendix C. DBpedia Dataset v.2015-10 Release Email

This appendix provides the email that was sent to the DBpedia community, which announces the release of the v.2015-10 DBpedia dataset. The author of this thesis is credited in the email for the work of interlink category mapping validation which is described in the case study in this thesis.

From: Markus Freudenberg <markus.freudenberg@gmail.com>
Date: 31 March 2016 at 12:51
Subject: [Dbpedia-discussion] ANN: DBpedia Version 2015-10 released
To: DBpedia <dbpedia-discussion@lists.sourceforge.net>, dbpedia-developers@list.sourceforge.net, dbpedia-ontology@list.sourceforge.net, semantic-web@w3.org, public-lod@w3.org, wikidata@lists.wikimedia.org, dbp-spotlight-users@lists.sourceforge.net

Hereby we announce the release of DBpedia 2015-10 (also known as: 2015 B).

This DBpedia release is based on updated Wikipedia dumps dating from October 2015 featuring a significantly expanded base of information as well as richer and (hopefully) cleaner data conforming to the DBpedia ontology.


Statistics
The English version of the DBpedia knowledge base currently describes 6.2M things of which 4.6M have abstracts, 955K have geo coordinates and 1.54M depictions. In total, 5M resources are classified in a consistent ontology and consists of 1.6M persons, 800K places (including 500K populated places), 480K works (including 133K music albums, 102K films and 20K video games), 267K organizations (including 66K companies and 52K educational institutions), 293K species and 5K diseases. The total number of resources in English DBpedia is 16.4M that, besides the 4.6M resources with abstracts, includes 1.3M skos concepts (categories), 7.1M redirect pages, 254K disambiguation pages and 1.6M intermediate nodes.

Altogether the DBpedia 2015-10 release consists of 8.8 billion (2015-04: 6.9 billion) pieces of information (RDF triples) out of which 1.1 billion (2015-04: 737 million) were extracted from the English edition of Wikipedia, 4.4 billion (2015-04: 3.8 billion) were extracted from other language editions and 3.2 billion (2015-04: 2.4 billion) from DBpedia Commons and Wikidata. In general we observed a significant growth in raw infobox and mapping-based statements of close to 10%.

Thorough statistics can be found on the DBpedia website and general information on the DBpedia datasets here.

Community
The DBpedia community added new classes and properties to the DBpedia ontology via the mappings wiki. The DBpedia 2015-10 ontology encompasses

• 739 classes (DBpedia 2015-04: 735)
• 1,099 object properties (DBpedia 2015-04: 1,098)
• 1,596 datatype properties (DBpedia 2015-04: 1,583)
• 132 specialized datatype properties (DBpedia 2015-04: 132)
• 407 owl:equivalentClass and 222 owl:equivalentProperty mappings external vocabularies (DBpedia 2015-04: 408 - 200)

The editors community of the mappings wiki also defined many new mappings from Wikipedia templates to DBpedia classes. For the DBpedia 2015-10 extraction, we used a total of 5553 template mappings (DBpedia 2015-04: 4317 mappings). For the first time the top language, gauged by number of mappings, is Dutch (606 mappings), surpassing the English community (600 mappings).

(Breaking) Changes
Lots of thanks to DBpedia ontology (dbo) and a full one with all types (ext). Both of these datasets use a typing system beyond the DBpedia ontology and we provide a subset, mapped to the datasets.

In addition, this release includes four new type datasets, although not included in the online sparql endpoint: 1) to identify and remove potentially wrong statements from the knowledge base. Information for 400,000 things that were formerly not typed. A similar algorithm described in Paulheim/Bizer 2014 for English DBpedia switched to IRIs from URIs. Some URIs will not resolve and we provide the "uri-same-as-iri" dataset for English to ease the transition. For more technical details on this issue read section 6 p. 19-23 (old but still valid).

The instance-types dataset is now split to two files:
- instance-types (containing only direct types)
- Instance-types-transitive containing the transitive types of a resource based on the DBpedia ontology

The mapping-based-properties file is now split in three (3) files:
- "geo-coordinates-mappingbased" that contains the coordinates originating from the mappings wiki. the "geo-coordinates" continues to provide the coordinates originating from the GeoExtractor
- "mappingbased-literals" that contains mapping based fact with literal values
- "mappingbased-objects" that contains mapping based fact with object values
- the "mappingbased-objects-disjoint-[domain|range]" are facts that are filtered out from the "mappingbased-objects" datasets as errors but are still provided

We added a new extractor for citation data.

All datasets are available in .ttl and .tql serialization (nt, nq dataset were neglected for reasons of redundancy and server capacity).

We are providing DBpedia as a Docker image. Dockerized-DBpedia: Creates and runs a Virtuoso Open Source instance preloaded with the latest DBpedia dataset inside a Docker container.

Starting with this release we provide extensive dataset metadata by adding DataIDs for all extracted languages to the respective language directories.

In addition we revamped the dataset table on the download page. It's created dynamically based on the DataIDs of all languages. Likewise the tables on the statistics page is now based on files providing information about all mapping languages.

From now on forward we also include the original Wikipedia dump files alongside the extracted datasets (pages_articles.xml.bz2).

A complete changelog can always be found in the git log

Upcoming Changes
- We are working to move away from the mappings wiki but we will have at least one more mapping sprint.
- We have some cool ideas for gsoc this year. Additional mentors are more than welcome:) 

Extended Type System to cover Articles without Infobox

Until the DBpedia 3.8 release, a concept was only assigned a type (like person or place) if the corresponding Wikipedia article contains an infobox indicating this type. Starting from the 3.9 release, we provide type statements for articles without infobox that are inferred based on the link structure within the DBpedia knowledge base using the algorithm described in Paulheim/Bizer 2014. For the new release, an improved version of the algorithm as run to produce type information for 400,000 things that were formerly not typed. A similar algorithm (presented in the same paper) was used to identify and remove potentially wrong statements from the knowledge base.

In addition, this release includes four new type datasets, although not included in the online sparql endpoint: 1) LHD datasets for English, German and Dutch and 2) DBTax for English. Both of these datasets use a typing system beyond the DBpedia ontology and we provide a subset, mapped to the DBpedia ontology (dbo) and a full one with all types (ext).

Credits
Lots of thanks to

- Markus Freudenberg (University of Leipzig / DBpedia Association) for taking over the whole release process and creating the revamped download page and statistics pages.
- Dimitris Kontokostas (University of Leipzig / DBpedia Association) for conveying his considerable knowledge of the extraction and release process.
- Volha Bryl (University of Mannheim / Springer) for their work on previous releases and their continuous support in this release.
- All editors that contributed to the DBpedia ontology mappings via the Mappings Wiki.
- The whole DBpedia Internationalization Committee for pushing the DBpedia internationalization forward.
- Heiko Paulheim (University of Mannheim) for re-running his algorithm to generate additional type statements for formerly untyped resources and identify and removed wrong statements.
- Václav Zeman and the whole LHD team (University of Prague) for their contribution of additional DBpedia types.
- Marco Fossati (FBK) for contributing the DBTax types.
- Alan Meehan (TCD) for performing a big external link cleanup.
- Aldo Gangemi (LIPN University, France & ISTC-CNR, Italy) for providing the links from DOLCE to DBpedia ontology.
- Kingsley Idehen, Patrick van Kleef, and Mitko Iliev (all OpenLink Software) for loading the new data set into the Virtuoso instance that provides 5-Star Linked Open Data publication and SPARQL Query Services.
• OpenLink Software (http://www.openlinksw.com/) altogether for providing the SPARQL Query Services and Linked Open Data publishing infrastructure for DBpedia in addition to their continuous infrastructure support.
• Ruben Verborgh from Ghent University – iMinds for publishing the dataset as Triple Pattern Fragments, and iMinds for sponsoring DBpedia’s Triple Pattern Fragments server.
• Ali Ismayilov (University of Bonn) for extending the DBpedia Wikidata dataset.
• Vladimir Alexiev (Ontotext) for leading a successful mapping and ontology clean up effort.
• All the GSoC students and mentors working directly or indirectly on the DBpedia release.
• Special thanks to members of the DBpedia Association, the AKSW and the department for Business Information Systems of the University of Leipzig.

The work on the DBpedia 2015-10 release was financially supported by the European Commission through the project ALIGNED – quality-centric, software and data engineering (http://aligned-project.eu/).
More information about DBpedia is found at http://dbpedia.org as well as in the new overview article about the project available at http://wiki.dbpedia.org/Publications.
Have fun with the new DBpedia 2015-10 release!
Cheers,
Markus Freudenberg, Dimitris Kontokostas, Sebastian Hellmann

---------------------------------------------------------------------------------------------------------------