Validating Interlinks between Linked Data Datasets with the SUMMR Methodology

Alan Meehan[1], Dimitris Kontokostas[2], Markus Freudenberg[2], Rob Brennan[1], Declan O'Sullivan[1]

[1] ADAPT Centre, Knowledge and Data Engineering Group, School of Computer Science and Statistics, Trinity College Dublin, Ireland
{meehanal,rob.brennan,declan.osullivan}@scss.tcd.ie
[2] AKSW, University of Leipzig, Computer Science, Leipzig, Germany
{Kontokostas,freudenberg}@informatik.uni-leipzing.de

Abstract. Linked Data datasets use interlinks to connect semantically similar resources across datasets. As datasets evolve, a resources locator can change which can cause interlinks that contain old resource locators, to no longer dereference and become invalid. Validating interlinks, through validating the resource locators within them, when a dataset has changed is important to ensure interlinks work as intended. In this paper we introduce the SPARQL Usage for Mapping Maintenance and Reuse (SUMMR) methodology. SUMMR is an approach for Mapping Maintenance and Reuse (MMR) that provides query templates which are based on standard SPARQL queries for MMR activities. This paper describes SUMMR and two experiments: a lab-based evaluation of SUMMR's mapping maintenance query templates and a deployment of SUMMR in the DBpedia v.2015-10 release to detect invalid interlinks. The lab-based evaluation involved detecting interlinks that have become invalid, due to changes in resource locators and the repair of the invalid interlinks. The results show that the SUMMR templates and approach can be used to effectively detect and repair invalid interlinks. SUMMR's query template for discovering invalid interlinks was applied to the DBpedia v.2015-10 release, which discovered 53,418 invalid interlinks in that release.

Keywords: Linked Data, Interlinks, Mapping Maintenance, Quality, DBpedia

1 Introduction

The Linked Open Data (LOD) cloud contains datasets covering a wide range of knowledge domains. Some datasets have overlapping knowledge domains which make them suitable for the establishment of interlinks between them. Interlinks are used to link semantically similar resources across datasets [1] in order to improve interoperability between them. However, Linked Data datasets are not static [2]. They evolve over time as new versions are periodically released or are updated in real time. This evolution can cause changes to the representation of resources which in turn can cause changes to a resources locator [3]. Changes to a resources locator can have an

effect on interlinks that references changed resources locators. This can cause interlinks to become invalid and no longer work due to a resource representation no longer dereferencing in the same manner as HTML "link rot". Discovering which interlinks have become invalid when a dataset evolves is important for the overall quality of the dataset. Some datasets have a large number of interlinks (ranging into the millions), making the discovery of invalid interlinks a time consuming task. So an approach for the detection of invalid interlinks would reduce the time and effort involved in their detection, where decisions can then be made about their repair or deletion, with the overall goal to improve dataset quality.

This paper focuses on the problem of discovering interlinks that have become invalid due to changes in the resource locators than an interlink references, where DBpedia is examined as our use-case. This involves discovering and retrieving the *source* and *target* resource locators of the interlinks and comparing them against their respective datasets to find if the resource locators still exist there.

The research question investigated in this paper is: to what extent can query templates, based on standard SPARQL queries, be used for discovering invalid interlinks? The appeal of relying upon standard SPARQL queries for this is that it can be performed on any standard SPARQL processor, which are widely deployed. This extends our previous work on SPARQL-based mapping management [4] which only supported mapping reuse, with a full methodology, SPARQL query templates and new experimental evidence.

The contribution of this paper is threefold. First is the SPARQL Usage for Mapping Maintenance and Reuse (SUMMR) methodology. Second is an initial lab-based evaluation of SUMMR's mapping maintenance query templates - to detect invalid interlinks and to repair them. Third is the evaluation of the application of SUMMR's query template for invalid interlink discovery, as part of the DBpedia v.2015-10 release process - in order to discover if any of those interlinks were invalid.

The remainder of this paper is structured as follows: In *Section 2*, we present existing approaches to mapping maintenance and reuse and we identify six individual tasks that need to be carried out for performing mapping maintenance and reuse. In *Section 3*, the SUMMR methodology is presented along with a prototype software implementation of it. *Section 4* then presents our initial lab-based experiment, which evaluated SUMMR's mapping maintenance query templates. In *Section 5* we present how SUMMR was applied to the DBpedia interlink management process and we also present the results from validating the interlinks from the DBpedia v.2015-10 release, with SUMMR. *Section 6* then finishes with conclusions and future work.

2 Linked Data Mapping Maintenance and Reuse

Semantic mappings are created between datasets to help reduce data heterogeneity and interoperability problems [5]. In this paper the definition of a mapping is referred to that of Euzenat and Shvaiko [5] as "…the oriented, or directed, version of an *alignment*…" which maps source and target resources in a certain way for a specific task. An *alignment* consists of source and target resources from two or more source

and target datasets, along with their *correspondences.* A *correspondence* then is the relation between the source and target resources (such as equivalent, not equivalent, more general than, less general than etc.).

Mapping maintenance involves the discovery and repair of invalid mappings. This is important as invalid mappings can lead to or produce incorrect results, causing interoperability and data quality problems in and among datasets. For the definition of mapping maintenance, we refer to the definition by Dos Reis et al. [6]: "[mapping maintenance is]…the task aiming to keep existing mappings in an updated and valid state, reflecting changes affecting KOS's (Knowledge Organization System) entities at evolution time". Note that the definition of a mapping by Dos Reis et al. differs from that of Euzenat and Shvaiko, where their definition of a mapping is the same as that of a correspondence (as described above). Given this difference between the two definitions, we feel that this definition of mapping maintenance is still appropriate as changes to *correspondences* (or mappings per Dos Reis et al. definition) can have an effect on *alignments* which can have effect on mappings. Given the definition of mapping maintenance above, invalid mappings are classed as mappings that are no longer correct due to changes in datasets that the mapping references.

Mapping reuse is important as the creation of new mappings is a difficult and time consuming task [7]. So being able to reuse previously existing mappings would help reduce the difficulties and effort it takes to create new mappings. We define mapping reuse as the use of existing mappings in three situations: a) specific mappings can be discovered and shared for direct reuse [8], b) information can be discovered from existing mappings to help with the creation of new mappings [9] and c) properties of existing mappings can be used in the creation of new mappings [1], [10].

The properties of a mapping refers to the details entailed within a mapping such as the source and target resources involved and any value transformations associated with a resource (for example converting miles to kilometers or concatenation of a person's first name and a last name). Meta-data can also be associated with a mapping and associated meta-data is also classed as a mapping property.

While the focus of this paper is on LOD interlinks, which connect resources across datasets, SUMMR is also concerned with managing vocabulary mappings [11]. Vocabulary mappings occur at the schema level and are used to transform instance data represented according to one vocabulary into another vocabulary. In this paper, when the broader word 'mapping' is used on its own to describe a SUMMR feature, it refers to both vocabulary mappings and interlinks.

2.1 Mapping Maintenance and Reuse Tasks

Given the scope of mapping maintenance and reuse (MMR) as described above, it is necessary to specify at a finer granularity the individual tasks that must be supported by a MMR approach. We do this by surveying the literature on existing MMR approaches (see next subsection) and identifying the key tasks that they support. For example, to perform both MMR activities, it must be possible to query mappings to discover and retrieve existing mappings and their individual properties. It also must be possible to detect if the source and target resources of mappings are still valid and

to alter a mapping for repair. Based on this survey, we have identified six key individual tasks that need to be carried out for MMR:

- **Task 1: Discover a mapping based on search criteria.** Specific mappings need to be discovered based on the searching of their properties.
- **Task 2: Discover the properties of a mapping.** Discovery of individual properties of specific mappings is necessary to discover information with existing mappings, where that information can then help with the creation of new mappings.
- **Task 3: Retrieve the entirety of a mapping**. Entire mappings need to be retrieved for sharing, direct reuse or alteration purposes.
- **Task 4: Retrieve individual properties of a mapping.** Individual properties of mappings need to be retrieved where they will be used in further steps, such as being used to create new mappings.
- **Task 5: Compare mapping properties to resources from a dataset.** This task is used to see if the source or target properties of a mapping are still valid in relation to the source and target datasets the mapping references.
- **Task 6: Alteration of an existing mapping.** Motivated by the need to change the properties of existing mappings for the purpose of repairing invalid mappings or deleting invalid mappings.

Note that not all of the above tasks may be useful by themselves as some of them are building blocks that are used in several activities. The tasks can also be combined to form additional activities, for example tasks 1, 4 and 5 would be combined to discover specific mappings and discover if any are invalid.

2.2 Existing Approaches

In this subsection we review notable existing approaches for both mapping maintenance and mapping reuse. We examine them in regards to the MMR tasks that they perform.

The Management of Ontology Mappings (MooM) framework [12], is a framework for mapping reuse. It provides storage for mappings that are enriched with meta-data from the Ontology Mapping, Management and Reuse (OM2R) [8] model. It provides a set of SPARQL queries to allow for the mappings to be discovered and retrieved. MooM is not concerned with the detection or repair of invalid mappings.

COMA++ [10] is an ontology mapping tool which includes a repository for storing mappings generated by the tool. It allows for interaction with the mappings for purposes such as editing, discovery of ‘mapping paths’ and sharing of mappings across other COMA++ applications. COMA++ lacks the ability for a user to perform precise querying and retrieval of mapping properties and it is not concerned with the detection of invalid mappings.

LinkLion [13] is an online repository for the storage, analysis and retrieval of interlinks between datasets. Users can submit interlinks to the repository, where they are represented according to the LinkLion vocabulary. LinkLion allows interlinks to be browsed and retrieved through a user interface or through a SPARQL endpoint. LinkLion is not concerned with the detection or repair of invalid mappings.

DSNotify [3] is a framework to detect change events of resources in datasets, with the aim to support the maintenance of interlinks. It monitors datasets and records changes to an event log. The event log contains the changed resources - along with *how*, *why* and *when* they changed. It then alerts a user that a change has occurred and the user can then use the event log to check if any interlinks have become invalid. DSNotify is not concerned with the reuse of mappings in anyway.

The research of *Khattak et al*. [14] proposes a mapping maintenance approach which detects changes to resources between two versions of a dataset. The changes are recorded to the Change History Ontology (CHO), which logs changes, the reason for change and the change agent. The CHO is then cross checked with mappings to find if they have become invalid. To repair invalid mappings, it proposes that ontology matchers should be used, where the changed resources (of a dataset) recorded in the CHO are used during matching, rather than an entire dataset. This way, less resources have to be matched overall, speeding up the repair process. This mapping maintenance approach is somewhat similar to that of DSNotify and as such, is not concerned with any mapping reuse activities.

DyKOSMap [15] is a framework for detecting and repairing mappings between biomedical datasets. It can detect changes in resources between two versions of a dataset and then discover invalid mappings effected by these changes. It then uses the invalid mappings, the detected dataset changes, ontology change patterns [16] and a designed heuristic to automatically attempt to repair invalid mappings. Potentially repaired mappings are outputted to be validated by a user. DyKOSMap is not concerned with mapping reuse so it does not provide functionality for a user to precisely query and retrieve mappings and their properties.

From reviewing the existing approaches, we have categorized which approaches perform which tasks in *Table 1* below. As can be seen, none of these approach perform all the tasks as none of these approach are concerned with both mapping maintenance and mapping reuse. We also included our approach, the SUMMR methodology, in the table to show that it is concerned with all the tasks.

Table 1. Existing approaches examined in regard to the mapping maintenacne and resue tasks they perform.'X' indicates a task is performed.

Task No.	***MooM***	***COMA++***	***LinkLion***	***DSNotify***	***Khattak et al***	***DyKOSMap***	***SUMMR***
1	X		X			X	X
2	X		X				X
3	X	X	X				X
4		X	X	X	X	X	X
5				X	X	X	X
6		X			X	X	X

3 The SUMMR Methodology

The SPARQL Usage for Mapping Maintenance and Reuse (SUMMR) methodology is an approach for MMR activities. SUMMR is an instantiation of a more general MMR approach that we propose. In the following two subsections we present a set of requirements for the general MMR approach and then present SUMMR - an instantiation of the general approach.

3.1 General MMR Approach Requirements

Below are a set of four requirements, which were derived from a review of existing approaches, for the general MMR approach. The general MMR approach relies on two features: 1) a mapping that represents mapping properties at a fine granularity and allows for meta-data annotation and 2) a declarative query-based approach for executing tasks over the mapping.

- **R1: A single, fine-grained mapping that allows for meta-data annotation must be used to represent mappings.** This will allow the approach to perform MMR activities over an entire range of mappings and the fine granularity and meta-data improves the discovery and retrieval of mappings [8] (see R2).
  R2: It must be possible to discover and retrieve existing mappings and their properties, ranging from *general* to *specific* search criteria. This is a necessary step in MMR activities. A fine grain mapping representation is useful here as it greater facilitates a *specific* search of the mappings.
- **R3: It must be possible to detect mappings that have become invalid due to changes in a referenced dataset.** This will highlight invalid mappings, where they can be used in further activities (such as the repair or deletion of the mapping).
- **R4: It must be possible to alter existing mappings and their properties.** This will allow existing mappings to be altered for purposes such as the repair of an invalid mapping by changing the invalid properties or deleting or retiring an invalid mappings that cannot be repaired.

3.2 SUMMR Design

Below we provide the design of SUMMR in relation to fulfilling the requirements in *Section 3.1*:

- For R1, the mappings are represented as SPARQL Centric Mappings (SCM) which we previously described in [4] and [17]. The SCM was originally designed to represent vocabulary mappings only but has now been expanded to also represent interlinks which allows for meta-data annotation in a similar manner to how BioPortal [18] and LinkLion [13] represent interlinks.
- For R2, the mappings will be stored in a triple-store with *SPARQL 1.1* and *SPARQL 1.1 Federated Query* compliance, so all mappings (local and remote) can be accessed. SPARQL SELECT/DESCRIBE queries are used to discover and retrieve mappings and the properties of mappings stored in the triple-store.

For R3, SPARQL SELECT queries are used to detect invalid mappings by detecting if the *source* and *target* resource locators in a mapping have changed by comparing these locators to the resource locators in the datasets the mapping references. Datasets can be accessed through the use of a federated SPARQL call, or a dataset dump can be loaded into a separate named graph in the local triple-store.

- For R4, SPARQL UPDATE queries are used to alter mappings. SPARQL INSERT queries are used to add new properties to existing mapping. SPARQL DELETE queries are used to delete properties from existing mappings and completely delete a mapping and its properties from the triple-store.

SUMMR provides a lightweight approach to MMR activities. Its aim is reduce the time and effort involved in 1) the detection of invalid mappings and 2) the creation of new mappings, through the reuse of existing ones. As can be seen from the design, SUMMR relies heavily on standard SPARQL queries and it provides a set of SPARQL query templates for performing MMR activities.

SPARQL provides important functionality for these activities, such as fine grain querying over mappings (given a sufficiently fine grain mapping [4]), federated calls to external datasets and the alteration of mappings through SPARQL UPDATE queries. Reliance on SPARQL also provides a standardized approach to MMR, since SPARQL is a W3C recommendation and is widely known and deployed and due to this, it is hoped will make SUMMR more appealing for adoption and deployment.

In relation to detecting invalid mappings, SUMMR does not detect all changes between two versions of a dataset [16] and then check which mappings have become invalid. Instead it provides a quick and easy to use detection of invalid mappings, by checking for changes of resource locators within the mappings by comparing these resource locators to the resource locators in the datasets the mapping references. This approach to detecting invalid mappings does have a limitation and it is related to when a resource changes [19] but the resources locator does not change. A situation where a resource may have changed semantically, to where the mapping is no longer semantically correct, but the resource locator has not changed is where this approach will fail to correctly discover an invalid mapping. Another situation is when a resource may have been changed location or be deleted but the old resource locator exists is where this approach will also fail to correctly discover an invalid mapping.

SUMMR is also not directly concerned with the *automatic repair* of mappings, however it can be used in conjunction with ontology matching systems [20]. These system can be used to produce new *alignments* between datasets. Based on these alignments then, new resource locators can then be used to update the SCM using the SUMMR template for repairing mappings.

3.3 SUMMR Query Templates

SUMMR provides six core query templates which were designed to support MMR activities. The activities that the templates support are based on activities that are supported by existing approaches for both mapping maintenance and mapping reuse but no existing approaches support all six. While there are six core query templates, each of these can vary slightly based on factors such as how general or specific the

mapping search criteria has to be or whether a federated SPARQL call is needed to access a remote dataset. *Table 2* below lists SUMMR's six query templates, indicating their category (reuse or maintenance) and which of the MMR tasks are involved in each query.

Table 2. SUMMR query templates

ID	*Template Purpose*	*Category*	*Tasks Involved*
QT1	Retrieve specific mappings for sharing	Reuse	1, 3
QT2	Discover information from existing mappings to aid in the creation of new mappings	Reuse	1, 2, 4
QT3	Discover mapping paths to aid in the creation of new mappings	Reuse	1, 4
QT4	Discover back links to aid in the creation of new mappings	Reuse	1, 4
QT5	Discover invalid mappings due to changes in datasets a mapping references	Maintenance	1, 4, 5
QT6	Repair of an invalid mapping	Maintenance	6

Below we provide an example of an interlink represented as a SCM and an example of a SUMMR query template - QT5.

```
01: @prefix sp: <http://spinrdf.org/sp#>.
02: @prefix gic: <http://www.scss.tcd.ie/~meehanal/gic>.
03: example:map001 a gic:Mapping;
04:       gic:wasCreatedBy example:person_x;
05:       gic:mapDescription "Linking X to Y"^^xsd:string;
06:       prov:generatedAtTime "2016-07-05"^^xsd:date;
07:       gic:hasRepresentation _:b1.
08: _:b1 a sp:Construct;
09:       sp:templates ([ sp:object lmdb:8;
10:       sp:predicate owl:sameAs;
11:       sp:subject dbpedia:Dr._Strangelove; ]);
12:       sp:where ();
13:       sp:text "CONSTRUCT
{<http://dbpedia.org/resource/Dr._Strangelove>
<http://www.w3.org/2002/07/owl#sameAs>
<http://data.linkedmdb.org/resource/film/8>}WHERE{}".
```

The SCM interlink above is concerned with linking a representation of the "Dr. Strangelove film" resource from the DBpedia dataset to the Linked Movie Data Base dataset. The mapping starts on line 03. Example mapping meta-data is on lines 04-06. SCMs use SPARQL CONSTRUCT queries (line 13) for mappings and use the SPIN [21] vocabulary to model the CONSTRUCT queries in a fine grain RDF syntax (lines 08-12).

```
01: SELECT DISTINCT ?invalid_mapping
02: WHERE {{?invalid_mapping gic:hasRepresentation ?rep.
03:  ?rep sp:templates/rdf:rest*/rdf:first/sp:subject ?s.
04:  FILTER NOT EXISTS {
05:   SERVICE **SOURCE_DATASET_SPARQL_ENDPOINT**
06:    { ?s ?p ?o. }} }
07:  UNION
08:  {?invalid_mapping gic:hasRepresentation ?rep1.
09:  ?rep1 sp:templates/rdf:rest*/rdf:first/sp:object ?t.
10:  FILTER NOT EXISTS {
11:   SERICE **TARGET_DATASET_SPARQL_ENDPOINT**
12:   { ?target ?p1 ?o1 .}} }}
```

This variation of QT5 above is concerned with the discovery of (non-specific) invalid interlinks. Placeholders in the queries are represented as **PLACEHOLDER**. The template first retrieves the *source* resource locator of the interlink on lines 02-03 (MMR Task 4 from *Section 2.1*) and then checks to see if that resource locator exists in the source dataset through a federated call on lines 04-06 (MMR Task 5). Then the template retrieves the *target* resource locator of the interlinks on lines 08-09 (MMR Task 4) and then checks to see if that resource locator exists in the target dataset through another federated call on lines 10-12 (MMR Task 5). If either of the source or target resource locators do not exist in the source and target datasets respectively, then a mapping is returned and classed as invalid.

3.4 SUMMR Interlink Validation Tool

We have developed a prototype tool that supports SUMMR-based interlink validation through facilitating the use of QT5, when given a source dataset and multiple target datasets. *Figure 1* below displays the operation of the tool.

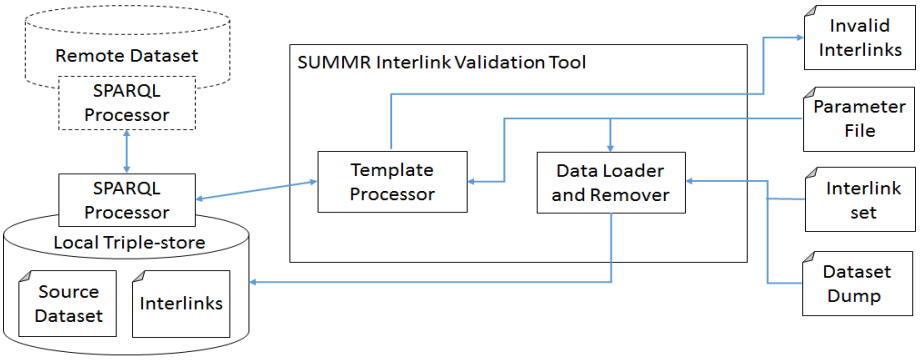

Fig. 1. Operation of the SUMMR Interlink Validation Tool. The arrows indicate the flow of information/data among the different components.

The *Parameter File* is where a user specifies the details about the interlinks that are to be validated (between the *Source Dataset* and remote external dataset). The parameters are: 1) the name of the external dataset; 2) the location of the interlinks in the

Local Triple-store (a URI of the named graph the interlinks are in) or the file path of a set of interlinks if the interlinks to be validated are stored in an external file; 3) an indicator specifying whether or not the external dataset resource locators of the interlinks are to be validated. If they are not to be validated, then only the local, source resource locators will be validated. If they are to be validated, then a user specifies how the resource locators will be validated against the respective external dataset. An external dataset can be accessed through a federated SPARQL query or a *Dataset Dump* file. So parameter four 4) 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 tool first checks where the interlinks are to be validated, if the interlinks 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 SUMMR QT5 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 form 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 resource locators 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 interlinks that are to be validated specified in the *Parameter File.*

The SUMMR Interlink Validation Tool was applied in the DBpedia interlink management process, for v.2015-10 DBpedia release (see *Section 5*).

4 Initial Evaluation

This section describes a small scale, controlled experiment to evaluate SUMMR's mapping maintenance query templates – QT5 and QT6. QT5 was evaluated in terms of its ability to discover invalid interlinks. QT6 was evaluated in terms of its ability to alter an invalid interlink for the purpose of repair. All data related to this experiment is available online[1].

The hypotheses for this experiment are:

H1: "SUMMR QT5 can be used to discover 100% of the invalid interlinks".

H2: "SUMMR QT6 can used to repair 100% of the invalid interlinks".

4.1 Datasets

Three datasets were used in this experiment: the *interlink dataset*, the *invalid interlink gold standard dataset* and the *repaired interlink gold standard dataset.* The two gold standard datasets were created by a postdoctoral researcher, with 7 years of expertise in Linked Data datasets, in Trinity College Dublin.

[1] http://www.scss.tcd.ie/~meehanal/Experiment3/

- **Interlink dataset:** This dataset consists of a set of 43 interlinks between the DBpedia and DailyMed datasets, which use the *owl:sameAs* property. These interlinks were created for and published along with version 3.3 of DBpedia. This interlink set was chosen as it contained a small number of interlinks - making it feasible for each of the interlinks to be manually verified (see *Invalid interlink gold standard dataset*). For this experiment, we represented these 43 interlink mappings as SCM representations. In the dataset, these mappings were assigned ID numbers 1 to 43.
- **Invalid interlink gold standard dataset:** This dataset contains the interlinks from the *interlink dataset* that are classed as invalid. These were determined through manually checking if the DBpedia and DailyMed resource locators, from each interlink, exist in *version 2015-04 of the DBpedia dataset* and a *dump of the DailyMed dataset* (that was collected via the DailyMed SPARQL endpoint on 02-September-2015) respectively. If a resource locator does not exist, then any interlink that contains that resource locator was marked as invalid. Also, for each invalid resource locator found, a *new locator* for the resource was manually searched for in *version 2015-04 of the DBpedia dataset* and the *dump of the DailyMed dataset* that could be used to repair the invalid interlink.
- **Repaired interlink gold standard dataset:** This dataset was created after the *invalid interlink gold standard dataset* was created. It contains the repaired version of the interlinks, that were classed as invalid from the *invalid interlink gold standard dataset.* They were manually repaired (where possible) with the *new locators* that were found in the *invalid interlink gold standard dataset.*

4.2 Method

A Jena Fuseki 1.3.0 triple-store and SPARQL server was used in this experiment for the execution of the SUMMR query templates.

Discovery of invalid interlinks:

1. The *interlink dataset* was loaded into a named graph in the triple-store.
2. The *DailyMed dataset dump* was loaded into a separate named graph in the triple-store.
3. A variation of SUMMR QT5 was used for the detection of invalid interlinks. In the template, the named graphs of the *interlink dataset* and the *DailyMed dataset dump* were specified. A federated SPARQL call to the DBpedia SPARQL endpoint[2] was also specified. The query template was then executed and the results of the invalid interlink mappings were recorded.
4. The results from the execution of QT5 was compared against the *invalid interlink gold standard dataset.*

Repair of invalid interlinks:

1. For each of the invalid interlinks discovered, a variation of SUMMR QT6 was prepared for its repair. In each template, the named graph of the *interlink dataset*, the

[2] http://dbpedia.org/sparql

invalid mapping identifier, the invalid resource locator and the new resource locator (used in place of the invalid resource locator) were specified. Each of these templates were executed.
2. The repaired interlinks were then compared against the *repaired interlink gold standard dataset*.

4.3 Results

Table 3 presents the results from comparing the execution results of QT5 against the *invalid interlink gold standard dataset*. The invalid interlink ID number is presented along with the recall and precision of the discovered invalid interlinks compared to the *invalid interlink gold standard dataset.*

Table 3. The discovered invalid interlink ID number presented with the recall and precision of the invalid interlinks discovered compared to the *invalid interlink gold standard*

Discovered Invalid Interlink ID Number	***Recall***	***Precision***
9, 11, 16, 21, 26, 28, 33	1.0	1.0

To evaluate QT6, the interlinks that were repaired with that template were compared to the *repaired interlink gold standard dataset*. The results are presented in *Table 4*.

Table 4. Repaired interlinks compared to *repaired interlink gold standard*, categorized as either *identical* or *not identical*.

Repaired Interlink Comparison Result	***Repaired Interlink ID Number***
Identical	9, 11, 21, 26, 33
Not Identical	16, 28

4.4 Discussion

As can be seen from the results in *Table 3*, QT5 achieved a recall and precision result of 1.0. The results in *Table 4* show that QT6 was able to repair 5 of the invalid interlinks, while 2 interlinks were not repaired. Note that interlink number 16 was deemed a mistake in the *interlink dataset*, so while it could be detected as invalid, it was not possible to repair it. Also no new resource locator was found to repair interlink mapping number 28.

These results indicate that SUMMR QT5 can be used to detect invalid interlinks, represented as SCMs. The results also show SUMMR QT6 can be used to repair the SCM representation of invalid interlinks (when a new resource locator has been found to replace the invalid resource locator).

It is important to note that in the *interlink dataset*, no cases were found where a resource changed semantically and the resource locator did not change or a resource was changed location or was deleted and the old resource locator still exists. If either of these cases did occur in the datasets, QT5 would *not* be able to correctly detect invalid mappings due to these cases.

While this experiment was quite small scale and controlled, it has provided evidence that SUMMR's mapping maintenance templates - based on standard SPARQL queries can be used for the detection and repair of invalid interlinks that are represented as SCMs. This gives us confidence that the templates will work effectively when applied larger scale problems - which we have done by applying variations of QT5, via the SUMMR Interlink Validation Tool, in the DBpedia interlink management process. This was done to validate the interlinks that are to be used in the DBpedia v.2015-10 release (see *Section 5*).

5 Applying SUMMR in the DBpedia Interlink Management Process

The DBpedia dataset consists of information that is extracted from Wikipedia articles, represented as RDF triples and described according to the DBpedia vocabulary. Wikipedia is a general encyclopedia, it covers a vast range of knowledge from multiple domains. DBpedia is one of the biggest hubs of the LOD cloud where the v.2015-04 release contained over 27 million interlinks to 36 external datasets.

The 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. The post-processing phase is concerned with tasks such as quality checking and the management of interlinks. As depicted in *Figure 2*, a DBpedia release has three interlink sources: a) a set of custom scripts for external sources like FreeBase, GeoNames and Yago, b) external interlink contributions through the DBpedia interlink repository[3] and c) interlinks copied from the previous release - when not in (a) or (b).

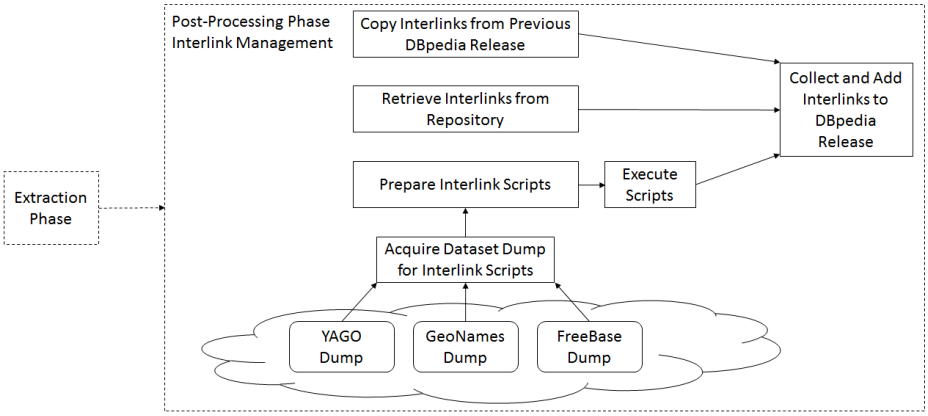

Fig. 2. DBpedia interlink management process

What is currently missing in the DBpedia interlink management process is interlink validation, to ensure that the interlinks to be published in the datasets are still valid. It

[3] https://github.com/dbpedia/dbpedia-links

is for this purpose that we apply SUMMR's query template for discovering invalid mappings (QT5) to the interlink management process.

5.1 Validating Interlinks for the 2015_10 Release of DBpedia

The application of SUMMR QT5, via the SUMMR Interlink Validation Tool, in the DBpedia interlink management process is depicted in *Figure 3.* The role of SUMMR QT5 in the process is to validate the interlinks that come from the interlink repository and the interlinks that are be copied over from the previous DBpedia release. Interlinks to the FreeBase, GeoNames and YAGO datasets are freshly generated and do not need to be validated. The interlinks from the interlink repository and the interlinks copied over need to be validated as they would have been created before the newest DBpedia release so they could be invalid in relation to that release.

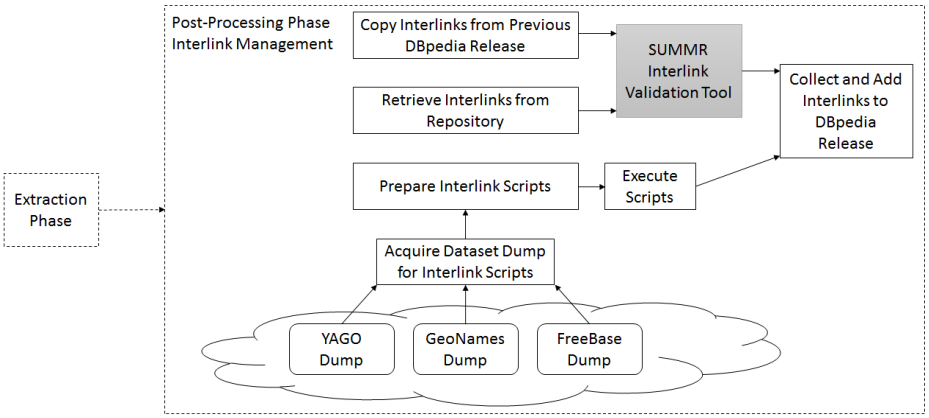

Fig. 3. DBpedia interlink management process with interlink validation provided by the SUMMR Interlink Validation Tool

The interlink validation involved checking that the DBpedia resource locators of the interlinks exist in the DBpedia v.2015-10 release. The external dataset resource locators of the interlinks can also be checked to ensure they exist in their respective dataset. If a DBpedia resource locator, from an interlink, does not exist in the DBpedia v.2015-10 release, and/or an external resource locator does not exist in its respective dataset, then that interlink is classed as invalid.

Below we describe the application of SUMMR QT5 to the interlinks that are to be used for the DBpedia v.2015-10 release - in order to discover if any of those interlinks are invalid.

Experimental Approach.

For the DBpedia v.2015-10 release, the interlinks from the previous DBpedia release (v.2015-04) were copied to the new release except for the interlinks to FreeBase, GeoNames, YAGO and Flickr Wrappr which were regenerated. Using SUMMR QT5, it checked if the DBpedia resource locators in the interlink sets exist in the DBpedia

v.2015-10 release. Some interlinks were also checked against their respective external datasets - when it was possible to access that dataset via a Federated SPARQL query.

A breaking change in the DBpedia v.2015-10 release was the switch from representing resources in the dataset with Uniform Resource Identifiers (URI) to 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 made a lot of existing DBpedia interlinks become invalid. To fix this problem, a separate dataset with URI-to-IRI mappings was published. Based on this change we validated the interlinks twice: a) without considering the URI-to-IRI mappings and b) taking the URI-to-IRI mappings into consideration. With this approach we measured the impact of the transition from URIs to IRIs had on the interlinks.

Steps:

The list of external datasets that DBpedia provides interlinks to can be found under the *"Links to other datasets"* section of the DBpedia v.2015-10 release download page[4]. Out of the 36 total external datasets, interlinks to 32 of these datasets consisting of 1,673,634 interlinks were validated with SUMMR QT5.

1. The DBpedia v.2015-10 release was loaded into a Virtuoso triple-store (Excluding the URI-to-IRI mappings).
2. The interlinks to the 32 external datasets were downloaded.
   Through the SUMMR Interlink Validation Tool, which generated variations of SUMMR QT5, the interlinks to the 32 external datasets were validated. All of the interlink sets were checked against the DBpedia v.2015-10 release. Seven external datasets were able to be accessed through a Federated SPARQL query and the interlinks related to these seven datasets were checked against them respectively. These seven datasets were: *CORDIS*, *DailyMed*, *DrugBank*, *EUNIS*, *Eurostat (Linked Statistics)*, *LinkedGeoData* and *OpenEI*.
3. The results from the execution of the tool were recorded.
4. The URI-to-IRI mappings were loaded into the triple-store.
5. Repeat of steps 3-4 to validate the interlinks where the URI-to-IRI mappings are considered.

Results.

Here we present the number of invalid interlinks found, for both situations when the URI-to-IRI mappings were not considered and when they were considered. 10.12% of the interlinks 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 interlinks for the DBpedia v.2015-10 release. Ultimately, 3.19% of the interlinks were found to

[4] http://wiki.dbpedia.org/Downloads2015-04

be invalid and these invalid interlinks were removed from the published v.2015-10 release. See *Table 5* below for a detailed breakdown of the invalid interlinks.

Table 5. Detailed breakdown of the interlinks validated by SUMMR QT5, categorized by external dataset name and indicating the number of interlinks per set, wether the interlink was vslidated against its external dataset and the number of invalid interlinks discovered when the URI-to-IRI mapping were not considered and when they were considered.

External Dataset	*No. of Interlinks*	*Externally Validated*	*No. of Invalid Interlinks (No URI-to-IRI Mappings)*	*No. of Invalid Interlinks (With URI-to-IRI Mappings)*
AmsterdamMuseum	627	No	74	10
BBCWildlife	444	No	4	1
Bookmashup	8,903	No	244	168
Bricklink	10,090	No	0	0
CIAFactbook	545	No	7	0
CORDIS	314	Yes	24	19
DailyMed	894	Yes	894	894
DBLP	196	No	5	4
DBTune	838	No	60	33
Diseasome	2,301	No	90	0
DrugBank	4,845	Yes	4,845	4,845
EUNIS	11,235	Yes	219	206
EuroStatLinkedStats	253	Yes	2	0
EuroStatWBSG	137	No	2	0
GADM	42,332	No	8	8
GeoSpecies	15,974	No	109	2
GHO	196	No	5	0
Gutenberg	2,510	No	163	5
ItalianPublicSchools	5,822	No	113	0
LinkedGeoData	103,633	Yes	27,194	4,559
LMDB	13,758	No	372	8
MusicBrainz	22,981	No	1,006	524
NewYorkTimes	9,678	No	35	6
OpenCyc	27,104	No	780	167
OpenEI	678	Yes	20	4
Revyu	6	No	0	0
SIDER	1,969	No	11	2
TCMGene	904	No	2	0
UMBEL	896,423	No	98,908	34,339
USCensus	12,592	No	2	0
WikiCompany	8,348	No	384	355
WordNet	467,101	No	33,809	7,265
TOTAL:	**1,673,634**		**169,391**	**53,418**
PERCENTAGE:			**10.12%**	**3.19%**

6 Conclusions and Future Work

In this paper we have presented the SUMMR methodology which is an approach for MMR activities using the SCM and standard SPARQL query templates.

Recall the research question investigated in this paper is: to what extent can query templates, based on standard SPARQL queries, be used for discovering invalid interlinks? The evidence obtained from our initial lab-based experiment indicates that SUMMR's mapping maintenance query templates (based on SPARQL queries) can be used to detect and repair invalid interlinks. The limitation of using standard SPARQL queries to detect invalid mappings means that invalid mappings will only be found due to changes in a resources locator - where the drawback of this was discussed in *Section 3.2*. However we still feel this approach is useful and our application of SUMMR QT5 to the DBpedia interlink management process has shown its usefulness. It has had a direct effect on the quality of the DBpedia v.2015-10 release by discovering that 53,418 (3.19%) of the interlinks were invalid.

Future work will involve extending our SUMMR software implementation to facilitate the additional functionality of SUMMR, namely the mapping reuse and mapping repair functionality.

Acknowledgements. 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 and funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 644055 (ALIGNED, http://www.aligned-project.eu).

References

1. Volz, J., Bizer, C., Gaedke, M. and Kobilarov, G. Discovering and maintaining links on the web of data. Springer Berlin Heidelberg, 2009.
2. Umbrich, J, Hausenblas, M., Hogan, A., Polleres, A. and Decker, S. "Towards dataset dynamics: Change frequency of linked open data sources." In *Proceedings of the Linked Data on the Web Workshop (LDOW2010)*, Raleigh, North Carolina, USA, April 27, 2010.
3. Popitsch, N. and Haslhofer, B. "DSNotify–A solution for event detection and link maintenance in dynamic datasets." Web Semantics: Science, Services and Agents on the World Wide Web 9, no. 3 (2011): 266-283.
4. Meehan, A., Brennan, R., & O'Sullivan, D. (2015, February). SPARQL based mapping management. In *Semantic Computing (ICSC), 2015 IEEE International Conference on* (pp. 456-459). IEEE.
5. Euzenat, J, and Shvaiko, P. *Ontology matching*. Vol. 18. Heidelberg: Springer, 2007.
6. Dos Reis, J. C., Pruski, C., and Reynaud-Delaître, C. "State-of-the-art on mapping maintenance and challenges towards a fully automatic approach." Expert Systems with Applications 42, no. 3 (2015): 1465-1478.
7. Falconer, S., and Noy, N. "Interactive techniques to support ontology matching." In Schema Matching and Mapping, pp. 29-51. Springer Berlin Heidelberg, 2011.

8. Thomas, H., Brennan, R. and O'Sullivan, D. "Using the OM 2 R Meta-Data Model for Ontology Mapping Reuse for the Ontology Alignment Challenge–a Case Study." *Ontology Matching* (2012): 61
9. Zhdanova, A. V. and Shvaiko, P. "Community-driven ontology matching." In *European Semantic Web Conference*, pp. 34-49. Springer Berlin Heidelberg, 2006.
10. Aumueller, D., Do, H., Massmann, S. and Rahm, E. "Schema and ontology matching with COMA++." In Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pp. 906-908. ACM, 2005.
11. Bizer, C. and Schultz, A. "The R2R Framework: Publishing and Discovering Mappings on the Web." In 1st International Workshop on Consuming Linked Data (COLD2010), Shanghai, China, November, 2010.
12. Thomas, H., Brennan, R. and O'Sullivan, D. "MooM--a prototype framework for management of ontology mappings." In Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on, pp. 548-555. IEEE, 2011.
13. Nentwig, M., Soru, T., Ngonga Ngomo, A, and Rahm, E. "LinkLion: A Link Repository for the Web of Data." In The Semantic Web: ESWC 2014 Satellite Events, pp. 439-443. Springer International Publishing, 2014.
14. Khattak, A. M., Pervez, Z., Khan, W. A., Khan, A. M., Latif, K. and Lee, S. Y. "Mapping evolution of dynamic web ontologies." Information Sciences (2015).
15. Dos Reis, J. C., Pruski, C., Da Silveira, M. and Reynaud-Delaître, C. "DyKOSMap: A framework for mapping adaptation between biomedical knowledge organization systems." Journal of biomedical informatics55 (2015): 153-173.
16. Hartung, M., Groß, A. and Rahm, E. "COnto–Diff: generation of complex evolution mappings for life science ontologies." Journal of biomedical informatics 46, 1 (2013): 15-32.
17. Meehan, A., Brennan, R., Lewis, D. and O'Sullivan, D. "Mapping Representation based on Meta-data and SPIN for Localization Workflows." In *Proceedings of the Second International Workshop on Semantic Web Enterprise Adoption and Best Practice at ESWC,* 2014.
18. Noy, N. F., Griffith, N. and Musen, M. A. "Collecting community-based mappings in an ontology repository." In *International Semantic Web Conference*, pp. 371-386. Springer Berlin Heidelberg, 2008.
19. Dos Reis, J. C., Pruski, C., Da Silveira, M. and Reynaud-Delaître, C. "Analyzing and supporting the mapping maintenance problem in biomedical knowledge organization systems." In Proc. of the Workshop on Semantic Interoperability in Medical Informatics collocated with the 9th Extended Semantic Web Conference, pp. 25-36. 2012.
20. Shvaiko, P. and Euzenat, J. "Ontology matching: state of the art and future challenges." *Knowledge and Data Engineering, IEEE Transactions on* 25, no. 1, 158-176, 2013.
21. Knublauch, H. SPIN SPARQL Inferencing Notation. *http://spinrdf.org/* [Accessed: 7-July-2016].