Detecting Inner-Ear Anatomical and Clinical Datasets in the Linked Open Data (LOD) Cloud

Muntazir Mehdi, Aftab Iqbal, Yasar Khan, Stefan Decker, and Ratnesh Sahay

Insight Centre for Data Analytics, NUI Galway, Ireland
`{firstname.lastname}@insight-centre.org`

Abstract. Linked Open Data (LOD) Cloud is a mesh of open datasets coming from different domains. Among these datasets, a notable amount of datasets belong to the life sciences domain linked together forming an interlinked "Life Sciences Linked Open Data (LSLOD) Cloud". One of the key challenges for data publishers is to identify and establish links between newly generated domain specific datasets and LSLOD Cloud. While a number of publishing tools exist for creating links from new to existing datasets, tools to detect domain-specific relevant datasets for linking purposes are missing. In this paper, we propose an extended technique for automatically identifying relevant datasets in LSLOD Cloud for inner-ear anatomical and clinical terminologies. We validate the proposed technique with experiments over the publicly accessible LSLOD Cloud using real-world terminologies and datasets provided by clinical organizations.

1 Introduction

The Linked Open Data (LOD) Cloud in its entirety is majorly composed of datasets published and updated by different publishers coming from academia, government organizations, online communities and companies alike. Most of the datasets within the LOD Cloud are accessible via at-least one SPARQL endpoint. The Datahub[1] or Mannheim Linked Data[2] catalogue lists such SPARQL endpoints available on the Web (though indeed some are offline or unreliable [1]). The LOD Cloud also comprises of 500 million links across datasets[3], following the fourth Linked Data principle: "*links to related data*". From the perspective of a consumer, these links allow for recursively discovering and navigating detailed information about related entities elsewhere on the Web. From the perspective of a publisher, links encourage modularity, where high-quality links (once in place) can reduce the amount of content they need to host. From the perspective of the Web, these links form the mesh upon which the Web of Data is based.

However, creating links with external LOD datasets is a challenging task for publishers. Addressing this challenge, a number of linking frameworks, such as

[1] `http://datahub.io/group/lodcloud`; l.a. 2015/08/15.
[2] `http://linkeddatacatalog.dws.informatik.uni-mannheim.de/dataset`; l.a. 2015/08/15.
[3] `http://lod-cloud.net/state/`; l.a. 2015/08/15.

Silk [11] and LIMES [7], have been proposed to help publishers link their local datasets to a remote LOD dataset. Given that there are now hundreds of remote datasets and many of them are black-boxes that do not describe their content [1], in practice it becomes a challenging task for a publisher to find potential datasets in order to setup links with the LOD Cloud. Currently, new publishers require knowledge of available LOD datasets. The publishers thus search and manually inspect the LOD datasets. In certain cases, the content of remote datasets are usually described using VoID[4], SPARQL 1.1 Service Descriptions, and specialized vocabulary (or ontology), which may help, but these are not available for many endpoints [1, 10].

This paper is based on our previous work of finding clinical trial datasets in the LOD Cloud [5, 6], specifically in LSLOD Cloud. Our main focus is on *Exact Literal Matching: a direct look-up of domain-specific keywords/terms, meaning an exact case-sensitive phrase match*. In this paper we extend our proposed "Multi Matching" algorithm [6] by (i) removing duplicate literal matching results and reducing the overall query execution time; and (ii) validating the extended algorithm on a real-life cochlea (inner-ear) datasets and terminologies. The approach presented in this paper will work for any remote dataset accessed via a SPARQL endpoint and will involve efficient look-ups, as opposed to inefficient post-filtering (REGEX Filters) and non-standard SPARQL (Full-text Search). Further, we present comparative evaluation of our previously proposed approaches and approach presented in this paper for a real-world use case. The rest of the paper is as follows: Section 2 presents our methodology and describes the extension of our previous work to query SPARQL endpoints in order to determine their relevance. Section 3 presents evaluation of our proposed approaches using different metrics, seeking relevant LSLOD datasets for interlinking. Section 4 discusses related work and Section 5 concludes this paper. Before moving to the next section, we introduce our motivating scenario involving data providers who wish to publish their corpora as Linked Data.

Motivating Scenario: Our work is conducted in the context of the SIFEM EU project[5], which aims at developing a semantic infrastructure interlinking an open source Finite Element Tool [2] with existing data, models and new knowledge for the multi-scale modeling of the inner-ear with regards to the sensorineural hearing loss. One of the core goals of the project is to publish biomedical datasets provided by the consortium partners as high-quality Linked Data.

Table 1: Example terms from datasets.

Category	**Example Terms**
Inner Ear Anatomy	afferent neuron, apex of cochlea, auditory nerve, basilar membrane of cochlea, bony labyrinth, hair cell, lower turn of the cochlea etc.
Clinical Variables	audiometry, bone vibrator, electrical noise, micro ct image, myogenic noise, rarefaction, rinneTest, surface electrode etc.

[4] `http://www.w3.org/TR/void/`; l.a. 2015/08/15.
[5] `http://www.sifem-project.eu/`; l.a. 2015/08/15.

The SIFEM terminologies[6], (Table 1 provides some examples of Terminologies) are used for the identification of potential LSLOD datasets. The life sciences community have been very active within the Linking Open Data movement. Of the 1,014 linked datasets that are contained in LOD Cloud, 83 datasets relate to life sciences, incorporating 3 billion facts and 191 million links in order to form LSLOD Cloud. The proposed method in this paper probe SPARQL endpoints of LSLOD datasets for detecting inner-ear anatomy and clinical datasets for linking.

2 Methodology: Detect Matching

The Detect Matching technique, presented in this paper, uses the Query Terms (QTerms) generated using our previously proposed tools given in [5]. The *QTerms* are used to probe SPARQL endpoints in order to find relevant LSLOD datasets. Each of the approach (Multi-Matching (μMatch) [6], and Detect Matching given in this paper) takes as input a set of *QTerms* and a list of URLs for different SPARQL endpoints (*SEndpoints*). We generate *SEndpoints* by logging URLs of all candidate SPARQL endpoints in a particular domain. For our use-case, we created a list of *SEndpoints* specifically for the life sciences domain. Therefore, we considered 35 SPARQL endpoints that are made publicly available by Bio2RDF[7].

Algorithm 1: DetMatch: Detect Matching Algorithm

```
Input: A QTerm, A language tag (lang-tag)
Output: A finite set of SPARQL endpoints (EPSet)
$SEndpoints$ := set of cataloged LSLOD SPARQL Endpoints;
for $EP$ in $SEndpoints$ do
    $values$ := ProcessTerm($QTerm$, lang-tag);
    $add$ := QueryAndLog($values$, EP);
    if $add$ = true then
        $EPSet := EPSet \cup \{EP\}$;
    end
end
return $EPSet$;
ProcessTerm($QTerm$, lang-tag)
    $PCase$ := Proper Case $QTerm$; $LCase$ := lower case $QTerm$; $UCase$ := UPPER CASE $QTerm$;
    $TypedQTerm$ := Typed $QTerm$;
    $TypedPCase$ := Typed $PCase$; $TypedLCase$ := Typed $LCase$; $TypedUCase$ := Typed $UCase$;
    //Type := http://www.w3.org/2001/XMLSchema#string
    $values$ := $QTerm$ ||$PCase$ ||$LCase$ ||$UCase$ ||
    $QTerm$@lang-tag ||$PCase$@lang-tag ||$LCase$@lang-tag ||$UCase$@lang-tag ||
    $TypedQTerm$ ||$TypedPCase$ ||$TypedLCase$ ||$TypedUCase$;
    return $values$;
QueryAndLog($values$, $EP$)
    $match$ := false;
    $Query$ := create SPARQL query;
    $Result$ := supply $Query$ to $EP$ and retrieve results;
    if $Result$ not empty then
        $match$ := true;
        log $Result$;
    end
    return $match$;
```

[6] `http://www.sifem-project.eu/sites/default/files/sigUploadedFiles/D5.4.1_Clinical%20Knowledge%20Documentation_final.pdf`; l.a. 2015/08/15.

[7] `http://download.bio2rdf.org/release/3/release.html`; l.a. 2015/08/15.

In [6], we presented Direct-Matching (DMatch) approach that creates a single SPARQL query per *QTerm* and Multi-Matching μMatch approach that creates eight different SPARQL queries for each *QTerm*. Although, (μMatch) covers possible variants for a *QTerm* that may exist at a specified SPARQL endpoint but it is a costly operation (as we observe later in the Section 3), because each *QTerm* is queried eight times on a specified SPARQL endpoint. A *QTerm* can be represented with and without a language tag in a specific LSLOD dataset. For example, consider a *QTerm* that appears as "cochlea" and "cochlea"@en on a particular LSLOD dataset, therefore, μMatch consider hits twice for the same *QTerm*. Our third approach is refined to exploit the VALUES clause provided in SPARQL 1.1, which can be used to provide a list of values that are used for query evaluation. By using VALUES keyword, we are able to supply all possible variants (e.g., Cochlea, cochlea, COCHLEA, Cochlea@en,Cochlea@en, cochlea@en, COCHLEA@en) for a *QTerm* in a single SPARQL query rather than executing multiple queries on a specified SPARQL endpoint (as shown in the Listing 1.1). Moreover, using the VALUES feature of SPARQL 1.1, we are able to eliminate the duplicate matched results for a single *QTerm*. The pseudo-code of DetMatch is shown in Algorithm 1.

```
SELECT DISTINCT ?s ?p WHERE {
        ?s ?p ?o .
        VALUES ?o {
        "Cochlea" "Cochlea" "cochlea" "COCHLEA"
        "Cochlea"@en "Cochlea"@en
        "cochlea"@en "COCHLEA"@en
        "Cochlea"^^http://../../XMLSchema#string
        ....... }
```

Listing 1.1: Example SPARQL query for DetMatch approach.

3 Evaluation

In Section 2, we presented three different approaches to probe public SPARQL endpoints for its relevance to a local dataset. In this section, we will compare the results of our proposed approaches by carrying out a series of experiments on the datasets provided by our clinical partners. All experiments were conducted on a computer running 64-bit Windows7 OS, with 4GB RAM and an Intel Core i5 (2.53 GHz) CPU. We use a MySQL RDBMS to store experimental data, including the endpoints to search and the results of successful queries. For experimentation, we consider a total of 220 clinical terminologies (CTerms) (some terminologies are exemplified in Table 1), which results in a total of 589 Query Terms (QTerms). And as mentioned in Section 2, we consider a total of 35 public endpoints from Bio2RDF. Queries are sent to the public endpoints over HTTP using the standard SPARQL protocol.

3.1 Results

We compare the results of our experiments based on three metrics: (i)**Matched Results (MR):**represents the total number of query results obtained for all terms against all SPARQL endpoints; (ii) **Matched Terms (MT):** represents the number of distinct terms for which non-empty results were found for at least one SPARQL endpoint; and (iii) **Matched Endpoints (ME):** represents the number of distinct endpoints for which some term generated a non-empty result [8].

A detailed comparison of MT, MR and ME based on the three approaches (DMatch, μMatch and DetMatch) is presented in Fig. 1–3 respectively. In general, we see a significant increase in the number of matched terms (cf. Fig. 1) when comparing DMatch approach against μMatch and DetMatch approaches. This is obvious due to the usage of different variants (i.e., lowercase, uppercase, proper case etc.) for each *QTerm* while probing SPARQL endpoints. The reason behind same number of matched terms for μMatch and DetMatch is because of using VALUES clause for DetMatch to supply all possible variants for each *QTerm* in a single SPARQL query whereas in μMatch, we executed multiple queries to cover all possible variants for each *QTerm*.

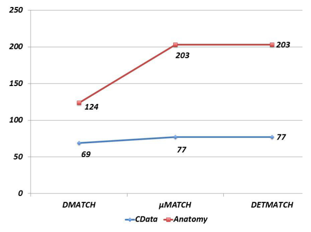

Fig. 1: Matched Terms (MT) comparison.

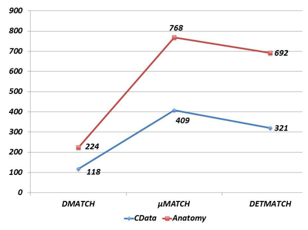

Fig. 2: Matched Results (MR) comparison.

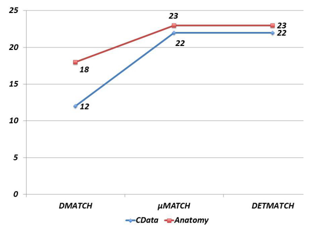

Fig. 3: Matched Endpoints (ME) comparison.

The significant increase in the number of matched terms (MT) subsequently increases the number of matched results (MR) for μMatch and DetMatch in contrast to DMatch approach (cf. Fig. 2). Comparing the matched results of μMatch against DetMatch (cf. Fig. 2), we see a decline in the later approach. The reason is the elimination of duplicate matched results for a *QTerm* using DetMatch approach, which is not handled in μMatch due to the usage of multiple queries. For example, consider an example term oxytocic that is contained in the BioPortal SPARQL endpoint[9]. This term is defined using dct:title and obo:exact_synonym predicates with the values "oxytocic"@en and "oxytocic"^^http://www.w3.org/2001/XMLSchema#string respectively. In the case of DetMatch, the term oxytocic is matched only once but in the case of μMatch, it is matched twice due to the usage of multiple variants for a *QTerm*. With respect to the number of SPARQL endpoints that are matched (cf. Fig. 3),

[8] To illustrate these metrics, consider an example where a set of two $QTerms = \{$ "afferent", "neuron" $\}$ is searched using either DMatch, μMatch or DetMatch on $SEndpoints = \{$ "BioPortal", "ClinicalTrials", "MeSH" $\}$, where the returned results contained 3 matches of "neuron" in "BioPortal", 2 matches in "ClinicalTrials" and 1 match in "MeSH". Similarly, the returned results for "afferent" had 0 matches in "BioPortal", "ClinicalTrials" and "MeSH". Then MR = 6, MT = 1 and ME = 3.

[9] `http://bioportal.bio2rdf.org/sparql`; l.a. 2015/08/15.

we observe that μMATCH and DETMATCH produce better results than DMATCH approach because of the usage of different possible variants for each *QTerm*.

Finally, we compare the query execution time for the three approaches (DMATCH, μMATCH and DETMATCH). The time (in seconds) is calculated based on the execution time of all *QTerms* (that are present in one dataset) on a specific SPARQL endpoint. Fig. 4 shows the average query execution time (based on 35 SPARQL endpoints that are considered) for the three different approaches on our datasets.

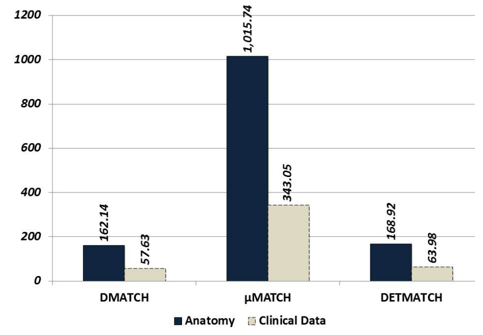

Fig. 4: Query execution time.

From Fig. 1 and Fig. 3, we see that μMATCH and DETMATCH have identified same number of matched terms as well as SPARQL endpoints but the overall execution time of μMATCH is much higher than that of DETMATCH (cf. Fig. 4). The reason is, μMATCH executes eight SPARQL queries per *QTerm*, however, DETMATCH uses all possible combinations for a *QTerm* in a single SPARQL query, thus reducing the overall query execution time. Similarly, it can also be noticed that the query execution time of DETMATCH (cf. Fig. 4) is slightly higher than that of DMATCH, with the amount of returned results as better as μMATCH (cf. Fig. 1 and Fig. 3).

In this section, we have presented results of three different approaches (DMATCH, μMATCH and DETMATCH) to probe SPARQL endpoints for its relevance to a local dataset. By evaluating the results based on three different metrics (MT, MR and ME) along with query execution time, we see that DETMATCH approach turns out to be the best option for discovering external LSLOD datasets that are potential targets for interlinking with local dataset. The reason is, DETMATCH has the ability to state all possible variants for each *QTerm* in a single SPARQL query, which lacks in the DMATCH approach and not cost effective in the case of μMATCH approach due to the $8\times$ query-load. From the publisher's perspective, if completeness of results is primarily important with less importance to the efficiency of discovery process (execution time), then we can say that μMATCH and DMATCH approaches are best suited for discovering public SPARQL endpoints that are potential targets for interlinking to a local dataset.

4 Related Work

This work aims to find relevant datasets on the LOD Cloud, LSLOD in particular. In past the majority of approaches taken in exploiting the generic and noisy LOD Cloud are performed in two categories (i) identify schema-level links (using *rdfs:subClassOf*, *owl:equivalentClass*) between class hierarchies among different RDF and OWL datasets; and (ii) identify instance-level links (using *owl:sameAs*) among different RDF and OWL datasets.

Instance-level links: Leme at al. [3] identifies datasets for interlinking and ranks them using probability measures based on a set of analyzed features. The proposed approach suggests links amongst different data sources using high level information, while schema- or instance-level information is not taken into account. Nikolov et al. [8, 9] propose an approach to identify relevant datasets for interlinking consisting of two steps: (1) searching relevant entities in other datasets using keywords; and (2) filtering irrelevant datasets based on semantic concept similarities using ontology matching techniques. We previously mentioned that there are a number of linking frameworks available for Linked Data, including Silk [11] and LIMES [7]. Both of these works provide a declarative language for guiding the creation of links between datasets based on predicates. However, both of these tools presume that a SPARQL endpoint for the target dataset is specified in the input. Our work addresses the prior step of identifying public endpoints of LSLOD datasets that are interesting to link to. Maali et al. [4] propose an extension of the Google Refine tool to curate and RDFize local datasets. The extension can help find legacy URIs for entities from target endpoints specified by the user. The authors propose using custom full-text search over SPARQL endpoints to find relevant URIs for keyword terms in the local dataset; e.g., using bif:contains over Virtuoso endpoints. However, they presume that the endpoints of interest are manually specified by the user. Other works have discussed the difficulties of discovering relevant SPARQL endpoints on the Web. For example, Buil Aranda et al. [1] note that few structured descriptions are available for endpoints. Paulheim and Hertling [10] also note that the discovery of endpoints is a difficult problem and propose methods to find endpoints given a URI of interest.

5 Conclusion

In this paper, we extend our approach of direct look-up of domain-specific keywords/terms in the LSLOD Cloud. We have been able to remove – compared to previous approach [5, 6] – duplicate literal matching results and reduce the overall query execution time. The extended approach is validated using the domain specific anatomical and clinical cochlea (inner-ear) terminologies. We envision that our approach will complement the state-of-the-arts in schema and instance linking tools for the LOD Cloud. Our approach will support linking frameworks to generate automated links (without requiring prior knowledge of LSLOD datasets), this in turn will help domain experts and scientists to explore domain-specific information contained in publicly available infrastructure like the LOD Cloud. The three algorithms (DMatch, μMatch and DetMatch) have certain advantages and trade-offs in context of query execution time, number of results, and results replication, therefore, we intentionally avoid any fit-for-all suggestion. Depending on different clinical contexts, i.e., set of input clinical terminologies, user may select one of the three algorithms or their combination targeting specific clinical scenario.

Acknowledgements*: This publication has emanated from research supported in part by the research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 and EU project SIFEM (contract Number 600933).*

References

1. C. B. Aranda, A. Hogan, J. Umbrich, and P.-Y. Vandenbussche. SPARQL Web-querying infrastructure: Ready for action? In *International Semantic Web Conference (2)*, pages 277–293. Springer, 2013.
2. V. Isailovic, M. Obradovic, D. Nikolic, I. Saveljic, and N. D. Filipovic. SIFEM project: Finite element modeling of the cochlea. In *13th IEEE International Conference on BioInformatics and BioEngineering, BIBE 2013, Chania, Greece, November 10-13, 2013*, pages 1–4, 2013.
3. L. A. P. P. Leme, G. R. Lopes, B. P. Nunes, M. A. Casanova, and S. Dietze. Identifying candidate datasets for data interlinking. In *Web Engineering*, pages 354–366. Springer, 2013.
4. F. Maali, R. Cyganiak, and V. Peristeras. Re-using cool URIs: Entity reconciliation against LOD hubs. In *Linked Data On the Web (LDOW) Workshop*. CEUR, 2011.
5. M. Mehdi, A. Iqbal, A. Hasnain, Y. Khan, S. Decker, and R. Sahay. Utilizing domain-specific keywords for discovering public SPARQL endpoints: a life-sciences use-case. In *Symposium on Applied Computing, SAC 2014, Gyeongju, Republic of Korea - March 24 - 28, 2014*, pages 333–335. ACM, 2014.
6. M. Mehdi, A. Iqbal, A. Hogan, A. Hasnain, Y. Khan, S. Decker, and R. Sahay. Discovering domain-specific public SPARQL endpoints: a life-sciences use-case. In *18th International Database Engineering & Applications Symposium, IDEAS 2014, Porto, Portugal, July 7-9, 2014*, pages 39–45. ACM, 2014.
7. A.-C. N. Ngomo and S. Auer. LIMES – a time-efficient approach for large-scale link discovery on the Web of Data. In T. Walsh, editor, *IJCAI*, pages 2312–2317. IJCAI/AAAI, 2011.
8. A. Nikolov and M. d'Aquin. Identifying relevant sources for data linking using a Semantic Web index. In *Linked Data On the Web (LDOW) Workshop*. CEUR, 2011.
9. A. Nikolov, M. d'Aquin, and E. Motta. What should I link to? Identifying relevant sources and classes for data linking. In *Joint International Semantic Technology Conference (JIST)*, pages 284–299. Springer, 2011.
10. H. Paulheim and S. Hertling. Discoverability of SPARQL endpoints in Linked Open Data. In *ISWC (Posters & Demos)*, pages 245–248. CEUR, 2013.
11. J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk - a link discovery framework for the Web of Data. In *Linked Data On the Web (LDOW) Workshop*. CEUR, 2009.