Proceedings of the Linked Data in Linguistics Workshop @ LREC2022, pages 52–60
Marseille, 24 June 2022

A Cheap and Dirty Cross-Lingual Linking Service in the Cloud

Christian Chiarcos[1,2], **Gilles Sérasset**[3]
[1]Applied Computational Linguistics, Goethe University Frankfurt, Germany
[2]Institute for Digital Humanities, University of Cologne, Germany
[3]GETALP, LIG, Univ. Grenoble Alpes, CNRS, Grenoble, France
chiarcos@cs.uni-frankfurt.de, gilles.serasset@imag.fr

Abstract

In this paper, we describe the application of Linguistic Linked Open Data (LLOD) technology for dynamic cross-lingual querying on demand. Whereas most related research is focusing on providing a static linking, i.e., cross-lingual inference, and then storing the resulting links, we demonstrate the application of the federation capabilities of SPARQL to perform lexical linking on the fly. In the end, we provide a baseline functionality that uses the connection of two web services – a SPARQL end point for multilingual lexical data and another SPARQL end point for querying an English language knowledge graph – in order to perform querying an English language knowledge graph using foreign language labels. We argue that, for low-resource languages where substantial native knowledge graphs are lacking, this functionality can be used to lower the language barrier by allowing to formulate cross-linguistically applicable queries mediated by a multilingual dictionary.

Keywords: Linguistic Linked Open Data, DBnary, DBpedia, cross-lingual querying

1. Introduction

Since its conception about a decade ago (Chiarcos et al., 2011), Linguistic Linked Open Data (LLOD) technology has begun to establish itself in the areas of language technology, linguistics and lexicography, most notably demonstrated by the development of a Linguistic Linked Open Data cloud,[1] and its increasing degree of maturity is demonstrated in a number of collected volumes, e.g., (Pareja-Lora et al., 2020), as well as a designated monography (Cimiano et al., 2020) that summarizes the state of the art in the field. As already observed by (Chiarcos et al., 2013), Linguistic Linked Open Data (LLOD) technology has a number of benefits in its application to language resources and language technology: The use of web standards such as RDF and SPARQL, as well HTTP-resolvable URIs for identifying and referring to content elements allows to establish links between resources published on the web of data, and this *linkability* entails advantages with respect to representation and modelling (graphs can represent any linguistic data structure), structural and conceptual interoperability (generic data structures, shared vocabularies, uniform access protocol), federation (querying over distributed data), dynamicity (access remote resources at query time) and the availability of a mature technical ecosystem for language technology.

In this context, especially the field of lexical resources has flourished, mostly due to the establishment and wide-spread adaptation of the OntoLex vocabulary[2] that expanded from its initial field of application from ontology lexicalization and the addition of linguistic information to general-purpose knowledge graphs to become a general community standard for machine-readable dictionaries on the web of data. With an increasing number of lexical data sets available as Linked Data or in RDF over the web, interest in *lexical* linking has been on the rise in the past years, e.g., in the Question-Answering over Linked Data challenges (QALD, since 2011)[3] or in the more series of Shared Tasks on Translation Inference Across Dictionaries (TIAD, since 2017).[4]

However, as far as lexical inference is concerned, all results we are aware of, be it on translation inference or the enrichment with multilingual labels, are concerned with precompiling links which are afterwards stored and distributed as novel or along with existing data sets. Curiously, the benefit of dynamicity, although being emphasized throughout the entire history of LLOD (Chiarcos et al., 2013; Cimiano et al., 2020), does not seem to have been explored for lexical data or cross-lingual linking, so far. We assume that this is mostly due to the fact that it is taken for granted, however, concrete applications of dynamic linking don't seem to ever have been brought forward.

With this paper, we aim to address this apparent gap. We describe the conjoint application of two web services, taking advantage of the federation capabilities of SPARQL to perform cross-lingual linking on the fly and thereby to enable querying over an English language knowledge graph using foreign language labels. Although this method is constrained by runtime considerations and thus largely restricted to string matching,[5]

[1] http://linguistic-lod.org
[2] https://www.w3.org/2016/05/ontolex/
[3] http://qald.aksw.org
[4] https://tiad2017.wordpress.com
[5] More advanced methods for lexical linking evaluate the wider lexicographical context, e.g., the number of pivot words that connect translation candidates (Lanau-Coronas and Gracia, 2020), but these are time-consuming analyses

we show that it provides a baseline functionality to enable the querying of knowledge graphs using foreign language query terms (words, or labels). This functionality, despite the noise it may introduce, is a valuable, and practically relevant option for low-resource languages for which no substantial knowledge graphs or ontologies exist, but whose speakers can then, for example, consult the English DBpedia in their own language.

We demonstrate this for two datasets and their associated web services (SPARQL end points): DBnary (Gilles Sérasset, 2012), described by Sérasset (2015), a machine-readable edition of Wiktionary data in RDF, and DBpedia (Auer et al., 2007), a machine-readable edition of Wikipedia data in RDF. However, instead of DBnary, any dictionary server could be used (e.g., the Apertium dictionaries hosted at UPM, (Gracia et al., 2018)), and instead of DBpedia, any knowledge graph (say, YAGO, (Suchanek et al., 2007)).

Overall, this paper is structured as follows: Sect. 2 discusses fundamentals of LLOD and RDF technology, Sect. 3 describes the use case. Then, Sect. 4 shows how we query DBpedia and DBnary. Finally Sect. 5 shows how we wrap up everything in one single federated query.

2. Linguistic Linked Open Data

As researcher specialised in Linguistic Linked Open Data, we are regularly faced with questions from other NLP or CL researchers on the advantages and drawback of using Semantic Web technologies to model, store or serve linguistic data. These questions are indeed justified as the Semantic Web approach seams to incur a steep learning curve and also may incur a higher workload on the resource publisher than on the resource consumer.

2.1. A Tree is a Graph, but a Graph is not Necessarily a Tree

Resource Description Format is the ground basis of the representation of Linked Open Datasets. RDF is not a language, but rather an abstract format that can be expressed using several syntax (one of which being indeed based on XML). RDF data is interpreted as a directed graph where (almost) each node has a name (an URI) and each arc is labelled using a relation name (also an URI).

As the data format is interpreted as a graph it's representation power is strictly higher than the representation power of a tree. As most existing Linguistic resources heavily use XML and seldom use XPaths/Xpointer to go beyond the basic XML tree structure, this leads to potentially more natural modelings for linguistic data.

One may argue that the OntoLex core vocabulary (McCrae et al., 2017) may mostly be viewed as a tree structure where root is the Lexicon, with LexicalEntry as children and Forms and LexicalSenses as grand-children. But you are able to provide additional information from other OntoLex vocabularies (e.g., OntoLex-VarTrans for Variation and Translation). For example, you can model a set of LexicalEntries/Forms/LexicalSenses using both the OntoLex model and give it additional structure with OntoLex-Lexicog (Bosque-Gil et al., 2017). OntoLex core will allow you to describe your LexicalEntries and LexicalSenses (and their relation to other ontologies or knowledge graphs) in a flat structure, while Lexicog model will allow you to precisely model the hierarchy of LexicalEntries/LexicalSenses as it was described in your original lexicon. As most of the nodes in both models are shared, the resulting structure may indeed be interpreted as 2 different trees covering the same node set.

2.2. No more Document Boundaries

By using URIs to name the nodes (and arcs) of the graph, each atomic part of your dataset is known outside of any file or document that may describe it. In essence, nodes in any RDF graph are globally defined and may be reused anywhere in the world. This is not the case for nodes of any XML files which may be shared between several documents only if the resource provider has given it a global name.

Moreover, the open world assumption clearly states that any document describing an entity can be complemented by any other source of information. Indeed, it is a common use case to describe the very same entity in different files or datasets. Among such use case are:

- by providing a set of relations between nodes from different datasets, one may link datasets together without being the producer of any of the linked datasets.
- in the DBnary dataset, the core of a dictionary (i.e. the lexical entries, word senses and canonical forms) is described in a dump file, while other dump files will complement lexical entries with all inflected forms or with translation links.

2.3. Standardizing Leads to Interoperability

The LLOD community is deeply involved in re-using common models for the publication of their linguistic data. As RDF allows for the extension of any vocabulary, it is easy for researcher to adopt a standard model even if some of its concepts are too coarse grained to be faithfully used for the description of a specific language.

One can easily refine the standard concepts by subclassing and provide a very detailed description of its lexicon. Then, consumers of the resource may be able to use the very fine-grain description or fallback to a coarser grained description for their specific use case.

that operate over large sets of complete dictionaries which are polynomial in time (over the size of the entire vocabulary). This is not an option here, as we require real-time performance, i.e., effectively linear lookup time.

2.4. Achieving Web Scale

When achieving the 5 stars of linked open data, each node in the LLOD graph has its own description available on the web (through it URI that is required to be resolvable through HTTP). Hence, consumers may be able to use LLOD data without necessarily having to import any dataset in their own database. Indeed any process that lacks knowledge on a specific entity may fetch it from the web.

Moreover, one can query different datasets through public SPARQL endpoints.

This means that the data available for your application goes far beyond what is available locally or in your own databases. In this paper, we show that it is possible to prototype a multilingual service without having any local database installed on premises.

3. Use Case: Cross-Lingual Querying of DBpedia

Our use case is the cross lingual querying of an ontology available in English. For this use case, we chose to query the English edition of DBpedia. Cross-linguality is achieved by querying DBnary, a multilingual dictionary available as Lexical Linked Open Data.

3.1. Knowledge graph: DBpedia

Since more than a decade, DBpedia is firmly established one of the most widely used general-purpose knowledge graphs in the web of data. At its core, it is automatically constructed from information provided in Wikipedia infoboxes. DBpedia started as a joint effort of researchers from Free University of Berlin and Leipzig University, Germany, in collaboration with OpenLink Software, and is now maintained by the University of Mannheim and Leipzig University. The first publicly available dataset has been made available in 2007 and published under the same license as the underlying Wikipedia information (CC-BY-SA), allowing others to reuse the dataset. DBpedia provides structured information extracted from Wikipedia pages and made available in a a uniform dataset which can be queried. As of June 2021, it contains over a trillion entities.

DBpedia has a broad scope of entities covering different areas of human knowledge. This allows external datasets to link to its concepts and has subsequently estalished DBpedia as a central hub in the web of data: The DBpedia dataset is interlinked with various other Open Data datasets on the Web, e.g., OpenCyc, UMBEL, GeoNames, MusicBrainz, CIA World Fact Book, DBLP, Project Gutenberg, Eurostat, UniProt, Bio2RDF, and US Census data.

DBpedia data can be queried via a public SPARQL endpoint under `https://dbpedia.org/sparql/`, which provides access to the underlying OpenLink Virtuoso data base.

3.2. Machine-readable dictionary: DBnary

The DBnary dataset (Gilles Sérasset, 2012) has grown steadily since its first description (Sérasset, 2012; Sérasset, 2015) and, at the time of writing, contains more than 275M relations describing 6.3M lexical entries in 22 languages. Its structure was originally based on lemon format but is now using the ontolex model.

The DBnary dataset now contains lexical data extracted from 22 wiktionary[6] language editions[7]. Up to now, DBnary used to only provide the wiktionary edition *endolexicon*, i.e. the subset of the wiktionary data that describe the language of the edition. That means that French language data is exclusively extracted from French language edition while English data was extracted from the English language edition. This choice was made so that data will achieve linguistic felicity as it is provided by the language's wiktionary community. Very recently DBnary is also providing the Wiktionary edition *exolexica*, i.e. all the lexical entries that do not belong to the edition's language. This means that many more languages may be described, but usually with a coarser grained description.

Translations are represented using an adhoc vocabulary based on the `dbnary:Translation` class which encodes a single translation from one of the extracted lexicon (endolex) to a target language. The translation entity is linked to a source lexical entry, but the target of the translation is encoded as a string, along with an entity representing the target language. Figure1 shows an example of such a translation.

```
fra:__tr_aze_1_animal__nom__1
    rdf:type dbnary:Translation ;
    dbnary:isTranslationOf
        fra:animal__nom__1 ;
    dbnary:targetLanguage lexvo:aze ;
    dbnary:writtenForm "heyvan"@az .
```

Figure 1: An example French to Azeri translation.

At the time of writing, the DBnary dataset contains 8.6M[8] such translations accounting for 22 source languages and 4396 different target languages. These number are constantly evolving as the DBnary dataset is extracted from Wiktionary everytime a dump is made available (i.e. twice a month). They should be compared with the 2.8M translations that where available in 2015. This changes in the datasets and in the whole LLOD cloud fully justify the use of dynamic approaches to lexical inferences.

[6] `http://wiktionary.org/`

[7] A language edition of wiktionary correspond to a site managed by its own community (e.g. `http://en.wiktionary.org` for English or `http://fr.wiktionary.org` for French). A language edition contains lexical entries in all possible language, with a description in the edition language.

[8] Exactly $8,619,352$ in the 20220401 semi-monthly extract from 1st April 2022, growing around 0.5% every month.

The DBnary dataset chose not to use ontolex `vartrans` (Bosque-Gil et al., 2015) by default as it is designed to link existing lexical entries through translation relations. In the case of DBnary, we do not have lexical entries in all the target languages and we chose not to adopt the LexVo (de Melo, 2015) attitude consisting in crafting a URI for every term in a language, as we are not guarantied that the value of a translation would qualify as a legitimate lexical entry in the target language (indeed, some translations are sometimes inflected forms or explanations rather than fully legitimate terms).
Note that translations from/to the 22 extracted languages are additionally represented using `vartrans` when the translation string can be linked to a lexical entry for the correct Part Of Speech, provided that there are no homonymy in the target language.
DBnary data can be queried via a public SPARQL endpoint under `http://kaiko.getalp.org/sparql`. Like DBpedia, this operates over an OpenLink Virtuoso data base.

4. Querying one end point at a time

4.1. SPARQL

In its current version 1.1, the SPARQL Protocol and RDF Query Language (SPARQL) [9] provides a standard for querying and manipulating RDF graph data over the Web or in an RDF store. SPARQL 1.1 defines a query language, result formats, update language, protocol and web service specifications. Features that set it apart from general query languages for graph data in general include query federation (accessing and integrating data from multiple remote end points at query time), entailment regimes (the possibility to infer implicit statements from an ontology associated with the data) as well as its orientation towards processing RDF data, i.e., a generic directed labelled multigraph characterized by using URIs (rather than internal IDs or strings) to denote nodes and edges, which can be serialized in or read from numerous formats (including, but not limited to, XML (Beckett and McBride, 2004), (X)HTML (Adida et al., 2008), JSON (Sporny et al., 2014), CSV (Ermilov et al., 2013), RDBMS (Dimou et al., 2014), as well as native RDF sources represented by Turtle (Beckett et al., 2014), HDT (Fernández et al., 2013), RDF-Thrift (Käbisch et al., 2015) or web services). If an RDF data set uses URIs that resolve via the HTTP prototocol to other RDF data, this constitutes Linked Data (Bizer et al., 2011), and linked data technology can been used to develop and to refer to widely used standards and community standards such as for knowledge graphs and ontologies – e.g., SKOS (Miles and Bechhofer, 2009), RDFS (McBride, 2004) and OWL (Antoniou and Harmelen, 2004) – as well as lexical data and other linguistic information – e.g., using SKOS-XL (Miles and Bechhofer, 2009), LexInfo (Cimiano et al., 2011), and OntoLex-Lemon (McCrae et al., 2017). With shared vocabularies described by resolvable URIs, Linked Data provides explicit, machine-readable semantics for its data structures that can be consulted at query time, and beyond shared vocabularies, the federation mechanism allows to also consult, retrieve and integrate information from different data providers.
Here, we focus on aspects of querying, the following section illlustrates federated search. In general, a SPARQL select query consists of a number of key words, including `PREFIX` (namespace declarations), `SELECT` (query operator), `FROM` (data source), and `WHERE` (graph pattern). The `WHERE` block contains the actual query, expressed using Turtle-style statements extended with variables, XPath-style functions for filtering and binding and operators such as conjunction (`.`), grouping (`{...}`), disjunction (`UNION`), negation (`MINUS`), optional statements (`OPTIONAL`), as well as the possibility to include embedded `SELECT` statements and to address named data sources (`GRAPH`) and remote end points (`SERVICE`). SPARQL contains a number of extensions over this basic model, including the possibility to not only query for individual statements, but also to formulate complex patterns over sequences of statements by means of SPARQL property paths.

4.2. Relations: DBpedia

For illustrating a general-purpose queries against a knowledge graph, imagine a simple question-answering setup in which we want to consult DBpedia to return (a human-readable label for) the type of entity a user enters. Say, for the query 'What is a horse?' (or, more briefly, 'horse'), we expect it to return 'animal'. The query itself, shown in figure 2, needs to be constructed on the basis of the RDF vocabularies used in DBpedia, but it contains a variable part with the actual search term, here, `"horse"@en`, i.e., *horse* in English:[10]
This query can be executed against the DBpedia end point.[11] It has a number of pecularities, so, the search term must be upper cased, also, we eliminate technical (W3C) terms that just describe the data model, and finally, we want to restrict the results (`?category`) to English labels. If these things are being respected, this query returns *animal* and *personal function*[12].
Another possible strategy is, instead of returning an English language label, to parse the local name of the URI

[9] `https://www.w3.org/TR/sparql11-overview/`

[10] Note that from future queries, we omit prefix declarations for reasons of space. All of the namespaces used can be retrieved using `http://prefix.cc/`. For example, `http://prefix.cc/dbp` will return `http://dbpedia.org/property/` for the namespace `dbp:`.

[11] `https://dbpedia.org/sparql`

[12] The latter value comes as a surprise, but it possibly comes from an logical inference based on the fact that 4 persons had "Horse" as their "function" in DBpedia

```
SELECT distinct ?category
WHERE {
  ?a rdfs:label "Horse"@en.
  ?a rdf:type ?type.
  FILTER(!strstarts(str(?type),
            'http://www.w3.org'))
  ?type rdfs:label ?category.
  FILTER(lang(?category)="en")
} LIMIT 10
```

Figure 2: Querying all categories an article labelled "Horse"@en belongs to.

as illustrated in Figure 3.

```
SELECT distinct *
WHERE {
  ?a rdfs:label "Horse"@en.
  ?a rdf:type ?type.
  FILTER(!strstarts(str(?type),
           'http://www.w3.org'))
  BIND(replace(str(?type),".*[/#]","")
         as ?localname)
  BIND(lcase(replace(?localname,
               "([a-z])([A-Z])","$1 $2"))
          as ?category)
}
```

Figure 3: Getting the name of the category by lower casing its URI localname.

As we cannot guarantee that URIs are not human-readable or in any particular language, the latter is usually discouraged, in this case, however, it retrieves additional categories that were not associated to an English label from an external vocabulary: *biological living object*, *eukaryotic cell* and *mammal*.

4.3. Translations: DBnary

There are several ways to query translations from the DBnary dataset.

4.3.1. Querying Translations Through Ontolex `vartrans`

Standard multilingual OntoLex modeled lexical datasets may be queried using `vartrans` ontolex extension. In `vartrans`, translations are represented by linking 2 lexical entries through `vartrans:translatableAs` relation. Figure 4 shows the corresponding query which looks for 2 lexical entries, related with this relation (in either direction), one of which is the term "Stadt"@de we want to translate.

This query will return *city*, *town*, *center*, *centre*, *stead* and *independent city* (where the two latter come from an inverse translation relation).

Such a query may be used on other datasets like Apertium or ACoLi and it indeed works on DBnary, but this will restrict our use case to the 22 language editions

```
SELECT DISTINCT * WHERE {
  ?source
    ontolex:canonicalForm/
      ontolex:writtenRep "Stadt"@de;
    (vartrans:translatableAs|
        ^vartrans:translatableAs)/
      ontolex:canonicalForm/
      ontolex:writtenRep ?translation.
    FILTER(lang(?translation)="en")
}
```

Figure 4: Querying translation from and to German term "Stadt" using `vartrans` modeling.

of DBnary (as these relations require a lexical entry at both end of the relation). This accounts for 3.5M available translation relations and 22 source/target languages while DBnary contains 8.6M translations to 4396 languages.

4.3.2. Querying Translations using DBnary's `Translation` class

As most translations in DBnary are encoded as `dbnary:Translation` instances. We can look for English translations of German entries or, in reverse, for German translations of English entries. Figure 5 retrieves translations for German *Pferd*.

```
SELECT distinct ?translation
WHERE {{
  ?t dbnary:isTranslationOf/
       ontolex:canonicalForm/
       ontolex:writtenRep "Pferd"@de;
     dbnary:writtenForm ?translation.
  } UNION {
  ?t dbnary:writtenForm "Pferd"@de;
     dbnary:isTranslationOf/
       ontolex:canonicalForm/
       ontolex:writtenRep ?translation.
  }
  FILTER(lang(?translation)="en")
}
```

Figure 5: Looking for translations from and to German *Pferd* using DBnary translations modeling.

This particular query returns *horse*, *equidae*, *knight*, *vaulting horse*, *vault*, *equine* and *horsy* and is equivalent to the preceding one for German to English, but it is now possible to query from languages that are not part of the 22 DBnary language edition. For instance, querying for translations of Romanian *cal* leads to *knight* and *horse*.

4.3.3. Querying translation using a pivot strategy

Previous strategies look for a translation instance from or to English. However, the DBnary dataset contains many more indirect translation links. The idea here is to find a translation relation from our query language to English by pivoting through one lexical entry of any

of the DBnary languages. Figure 6 shows such a query again for German term *Pferd*.

```
SELECT distinct ?translation
WHERE {
  ?entry ^dbnary:isTranslationOf/
          dbnary:writtenForm "Pferd"@de.
  ?t dbnary:isTranslationOf ?entry;
     dbnary:targetLanguage lexvo:eng;
     dbnary:writtenForm ?translation.
}
```

Figure 6: Translations of German *Pferd* pivoting on any lexical entry.

Executed against the DBnary SPARQL end point, this retrieves a total of 49 translations, while querying for Romanian *cal* leads to 46 translations.

5. Cheap and dirty cross-lingual querying by federated search

The idea of cheap and dirty cross-lingual querying is to use the functionalities illustrated above to translate labels from the user language to English, to extrapolate the expected DBpedia labels (by doing upper case conversion), to retrieve category labels from DBpedia and then to use DBnary to translate these back into the user language.

Note that we use German as an example only – and more precise results would be expected if we just query German DBpedia labels –, but this approach can be applied to *any* language for which lexical data in OntoLex is provided, and in particular, low resource languages. To a considerable extent, these are covered by DBnary, and as it is regularly updated from Wiktionary which is a crowd-sourced resource, its coverage is continuously increasing, but other portals provide OntoLex-compliant lexical data, as well, e.g., bilingual dictionaries from the Apertium project (Gracia et al., 2018), the GlobalWordNet family of resources (McCrae et al., 2021) or the ACoLi Dictionary Graph (Chiarcos et al., 2020).

In this case, both end points run on independent installations of the same database management system, however, it is important to note that no provider-specific technology is being used, but that we only rely on standardized, portable SPARQL 1.1 functionalities. In particular, this includes the keyword `SERVICE` which allows to consult an external SPARQL end point at query runtime.

Using the `SERVICE` keyword, it is possible to consult an external SPARQL end point (or another webservice) when running a local SPARQL query.[13] For example, as shown in figure 7, we can call DBpedia from DBnary, i.e., we first translate German to English, adjust the result to match the upper case convention of DBpedia labels, then query DBpedia and then translate the results back to German.

```
SELECT distinct ?result
       (count(distinct *) as ?confidence)
WHERE {
  ?entry ^dbnary:isTranslationOf/
          dbnary:writtenForm "Pferd"@de.
  ?t dbnary:isTranslationOf ?entry;
     dbnary:targetLanguage lexvo:eng;
     dbnary:writtenForm ?translation.

  BIND(concat(
         ucase(substr(?translation,0,1)),
         substr(?translation,2))
       as ?dbp_label)

  SERVICE <https://dbpedia.org/sparql> {
    ?a rdfs:label ?dbp_label.
    ?a rdf:type ?type.
    FILTER(!strstarts(str(?type),
            'http://www.w3.org'))
    ?type rdfs:label ?category.
    FILTER(lang(?category)="en")
  }

  ?t2 dbnary:isTranslationOf ?entry2;
      dbnary:targetLanguage lexvo:eng;
      dbnary:writtenForm ?category.
  ?entry2 dct:language lexvo:deu;
      ontolex:canonicalForm/
        ontolex:writtenRep ?result;
      a lexinfo:Noun
} ORDER BY desc(?confidence) asc(?result)
  LIMIT 10
```

Figure 7: Federated query calling DBpedia from DBnary after translations has been queried.

We return nominal concepts only, and with this particular query, we also calculate confidence, i.e., the number of paths (English translations, DBpedia concepts) that will lead to a particular translation. This is a very useful feature as the multitude of paths can lead to unexpected associations, and these can be detected as possible but unlikely.

Results of the query above are shown in Tab. 1. The top-level match is the expected result, *Tier* 'animal', *Couleur*, *Farbe*, *Farbton* 'color' and *Beere* 'berry' refer to types of horses designated by characteristics of their color, *Leut* 'people', *Mensch* 'human' and *Person* 'person' originate in the DBpedia concept *dpo:PersonalFunction* – this seems to reflect the sense of a 'workhorse' which can be metaphorically extended to people. It is less clear where *Chaot* 'slob' and *Gelähmter* 'paralyzed' originate from.

As this requires substantial aggregation, this is not really cheap, yet, but we can speed it up by restricting

[13] For security reasons and load balancing, this functionality may be disabled. It is, however, part of the SPARQL specification and should be supported by all SPARQL 1.1 compliant RDF stores.

result	confidence
"Tier"@de	271
"Couleur"@de	28
"Chaot"@de	22
"Gelähmter"@de	20
"Leut"@de	20
"Mensch"@de	20
"Person"@de	20
"Farbe"@de	14
"Farbton"@de	14
"Beere"@de	4

Table 1: Quick and dirty cross-lingual search: German category labels for German *Pferd* 'horse' retrieved from DBnary and the English DBpedia

the number of responses and suppressing aggregation as shown in figure 8

```
SELECT ?result WHERE {
  {SELECT DISTINCT ?translation
   WHERE {
     ?e ^dbnary:isTranslationOf/
         dbnary:writtenForm "Pferd"@de.
     ?t dbnary:isTranslationOf ?e;
        dbnary:targetLanguage lexvo:eng;
        dbnary:writtenForm ?translation.
   } LIMIT 3
  }

  BIND(concat(
         ucase(substr(?translation,0,1)),
         substr(?translation,2))
       as ?dbp_label)

 SERVICE <https://dbpedia.org/sparql> {
  SELECT DISTINCT
     ?dbp_label ?type ?category
  WHERE {
    ?a rdfs:label ?dbp_label.
    ?a rdf:type ?type.
    FILTER(!strstarts(str(?type),
            'http://www.w3.org'))
    ?type rdfs:label ?category.
    FILTER(lang(?category)="en")
  } LIMIT 2
 }

?t2 dbnary:isTranslationOf ?entry2;
    dbnary:targetLanguage lexvo:eng;
    dbnary:writtenForm ?category.
?entry2 dct:language lexvo:deu;
    ontolex:canonicalForm/
      ontolex:writtenRep ?result;
    a lexinfo:Noun
} LIMIT 1
```

Figure 8: Restricting the number of translation and suppressing aggregation.

It is to be noted that SPARQL query results are considered to be unsorted. Accordingly, the confidence we explicitly measured in the last query translates to a *probability* that the result of this query is correct. With *Tier* returned via 271 different paths in the last query from a total of 433 paths, and considering *Tier* to be the only correct response, the precision *for this particular query* would be at 62.5%.

6. Outlook

We have shown how two SPARQL webservices can be used in conjunction to perform cross-lingual search using search terms in one language, a knowledge graph in another, and producing results in the search language. We would like to note that this is not a novel functionality, but provided by standard SPARQL technology, albeit one which does not seem to have been documented in LLOD literature before. As such, our contribution is not so much innovative as it fills a gap in the current scientific documentation of LLOD practices and possibilities that can serve as a template for the development of future applications.

The main purpose of this submission is to show that SPARQL allows to stack web services and RDF resources in a meaningful, and specifically, to enable cheap and dirty cross-lingual querying. It's main advantages is that (1) it allows for crafting cheap mockups of cross-lingual services and (2) it fully uses dynamicity and the quality of the service will evolve with the quality/coverage of the resources available in the cloud.

It is clear that in this context, methods that compile static links are superior in quality,[14] but they require designated development time, substantial preprocessing and hosting of generated links, whereas the benefit of this method is that it is immediately applicable to *any* language for which a bilingual OntoLex dictionary can be found that either provides English translations or a link to another bilingual dictionary that does. At the time of writing, hundreds of such dictionaries are available online, e.g., from the OntoLex edition of PanLex (Kamholz et al., 2014) available from the ACoLi Dictionary Graph (Chiarcos et al., 2020).[15]

[14] Due to the trade-off between speed and quality, the comparative performance between static dictionary induction and dynamic methods is relatively hard to evaluate: Static methods are optimized towards managing and minimizing noise in translations, and they achieve this by aggregating confidence scores over the lexical content of pivot translations in larger lexical knowledge graphs. As shown in Tab. 1, aggregation is possible in our approach as well, but it comes at the price of processing speed, and for responses calculated on-the-fly, there are limitations as to how much context can be inspected. Our approach will produce optimal results (in terms of speed) if the number of pivot languages (or pivot translations) is limited. For static methods, where speed at query time is eliminated as a factor (i.e., reduced to a lookup), best results (in terms of quality) will be achieved if multiple pivot languages (or pivot translations per word) are available.

[15] https://github.com/acoli-repo/

Acknowledgements

The research described in this paper has been initiated in the context of the Cost Action *Nexus Linguarum. European Network for Web-Centered Linguistic Data Science* (CA18209). The work of the first author has been conducted in the context of the H2020 Research and Innovation Action *Prêt-à-LLOD. Ready-to-use Multilingual Linked Language Data for Knowledge Services across Sectors* (2019-2022, grant agreement 825182).

7. Bibliographical References

Adida, B., Birbeck, M., McCarron, S., and Pemberton, S. (2008). RDFa in XHTML: Syntax and processing. Technical report, W3C Recommendation.

Antoniou, G. and Harmelen, F. v. (2004). Web Ontology Language: OWL. In *Handbook on ontologies*, pages 67–92. Springer.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). DBpedia: A nucleus for a web of open data. In *The Semantic Web*, pages 722–735. Springer.

Beckett, D. and McBride, B. (2004). RDF/XML syntax specification (revised). Technical report, W3C recommendation.

Beckett, D., Berners-Lee, T., Prud'hommeaux, E., and Carothers, G. (2014). RDF 1.1 Turtle. *World Wide Web Consortium*, pages 18–31.

Bizer, C., Heath, T., and Berners-Lee, T. (2011). Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI global.

Bosque-Gil, J., Gracia, J., Aguado-de Cea, G., and Montiel-Ponsoda, E. (2015). Applying the OntoLex Model to a multilingual terminological resource. In Fabien Gandon, et al., editors, *The Semantic Web: ESWC 2015 Satellite Events*, pages 283–294, Cham. Springer International Publishing.

Bosque-Gil, J., Gracia, J., and Montiel-Ponsoda, E. (2017). Towards a Module for Lexicography in OntoLex. In *LDK Workshops*, pages 74–84.

Chiarcos, C., Hellmann, S., and Nordhoff, S. (2011). Towards a Linguistic Linked Open Data cloud: The Open Linguistics Working Group. *TAL Traitement Automatique des Langues*, 52(3):245–275.

Chiarcos, C., McCrae, J., Cimiano, P., and Fellbaum, C. (2013). Towards open data for linguistics: Linguistic linked data. In *New Trends of Research in Ontologies and Lexical Resources*, pages 7–25. Springer.

Chiarcos, C., Fäth, C., and Ionov, M. (2020). The ACoLi dictionary graph. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 3281–3290.

Cimiano, P., Buitelaar, P., McCrae, J., and Sintek, M. (2011). LexInfo: A declarative model for the lexicon-ontology interface. *Journal of Web Semantics*, 9(1):29–51.

Cimiano, P., Chiarcos, C., McCrae, J. P., and Gracia, J. (2020). *Linguistic Linked Data: Representation, Generation and Applications*. Springer Nature.

de Melo, G. (2015). Lexvo.org: Language-related information for the Linguistic Linked Data cloud. *Semantic Web*, 6(4):393–400, August.

Dimou, A., Vander Sande, M., Slepicka, J., Szekely, P., Mannens, E., Knoblock, C., and Van de Walle, R. (2014). Mapping hierarchical sources into RDF using the RML mapping language. In *2014 IEEE International Conference on Semantic Computing*, pages 151–158. IEEE.

Ermilov, I., Auer, S., and Stadler, C. (2013). CSV2RDF: User-driven CSV to RDF mass conversion framework. In *Proceedings of the ISEM*, volume 13, pages 04–06.

Fernández, J. D., Martínez-Prieto, M. A., Gutiérrez, C., Polleres, A., and Arias, M. (2013). Binary RDF representation for publication and exchange (HDT). *Journal of Web Semantics*, 19:22–41.

Gracia, J., Villegas, M., Gomez-Perez, A., and Bel, N. (2018). The Apertium bilingual dictionaries on the web of data. *Semantic Web*, 9(2):231–240.

Käbisch, S., Peintner, D., and Anicic, D. (2015). Standardized and efficient RDF encoding for constrained embedded networks. In *European Semantic Web Conference*, pages 437–452. Springer.

Kamholz, D., Pool, J., and Colowick, S. (2014). PanLex: Building a resource for panlingual lexical translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3145–3150.

Lanau-Coronas, M. and Gracia, J. (2020). Graph exploration and cross-lingual word embeddings for translation inference across dictionaries. In *Proceedings of the 2020 Globalex Workshop on Linked Lexicography*, pages 106–110.

McBride, B. (2004). The Resource Description Framework (RDF) and its vocabulary description language RDFS. In *Handbook on Ontologies*, pages 51–65. Springer.

McCrae, J. P., Bosque-Gil, J., Gracia, J., Buitelaar, P., and Cimiano, P. (2017). The Ontolex-Lemon model: development and applications. In *Proceedings of eLex 2017 conference*, pages 19–21.

McCrae, J. P., Goodman, M. W., Bond, F., Rademaker, A., Rudnicka, E., and Da Costa, L. M. (2021). The GlobalWordNet formats: Updates for 2020. In *Proceedings of the 11th Global Wordnet Conference*, pages 91–99.

Miles, A. and Bechhofer, S. (2009). SKOS simple knowledge organization system reference. *W3C recommendation*.

Pareja-Lora, A., Lust, B., Blume, M., and Chiarcos, C. (2020). *Development of Linguistic Linked Open*

acoli-dicts

Data Resources for Collaborative Data-Intensive Research in the Language Sciences. The MIT Press.

Sérasset, G. (2012). DBnary: Wiktionary as a LMF based Multilingual RDF network. In *Language Resources and Evaluation Conference, LREC 2012*, Istanbul, Turkey, May.

Sérasset, G. (2015). DBnary: Wiktionary as a Lemon-Based Multilingual Lexical Resource in RDF. *Semantic Web – Interoperability, Usability, Applicability*, 6(4):355–361.

Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., and Lindström, N. (2014). JSON-LD 1.0. Technical report, W3C recommendation.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

8. Language Resource References

Gilles Sérasset. (2012). *DBnary: Wiktionary as a Lemon-Based Multilingual Lexical Resource in RDF*. Univ. Grenoble Alpes, ISLRN 023-163-901-149-4.