ExConQuer: Lowering barriers to RDF and Linked Data re-use

Editor(s): Werner Kuhn, University of California, Santa Barbara, USA
Solicited review(s): Martin Tomko, The University of Melbourne, Australia; Andrea Ballatore, Birkbeck University of London, UK; Tomi Kauppinen, Aalto University School of Science, Finland

Judie Attard [*], Fabrizio Orlandi and Sören Auer
Enterprise Information Systems, University of Bonn, Regina-Pacis-Weg 3, 53113 Bonn, Germany
E-mail: attard@iai.uni-bonn.de, orlandi@iai.uni-bonn.de, auer@cs.uni-bonn.de

Abstract.
A major obstacle to the wider use of semantic technology is the perceived complexity of RDF data by stakeholders who are not familiar with the Linked Data paradigm, or are otherwise unaware of a dataset's underlying schema. In order to help overcome this barrier, we propose the ExConQuer Framework (Explore, Convert, and Query Framework) as a set of tools that preserve the semantic richness of the data model while catering for simplified and workable views of the data. Through the available tools users are able to explore and query linked open datasets without requiring any knowledge of SPARQL or the datasets' underlying schema. Moreover, executed queries are persisted so that they can be easily explored and re-used, and even edited. Therefore, with this framework we attempt to target the evident niche in existing tools that are intended to be used by non-experts to consume Linked Data to its full potential.

Keywords: Linked Data, Consumption Framework, Publishing

1. Introduction

The radical advances in technology, particularly though the advancement of the World Wide Web, have created new means to share knowledge. However, although barriers to information access have been lowered through various means (e.g. hypertext links, web search engines, REST APIs), accessibility to raw data was only afforded the same importance in recent years [2]. One of the catalysts for this change is the increasing adoption of Linked Data practices, as indicated by the extraordinary growth in the Linked Open Data Cloud's[1] volume from 2007 to 2014, as well as the number of triples continuously crawled by the LOD Laundromat[2]. Whereas raw data used to be published in formats such as CSV, which need metadata to be interpretable, the implementation of Linked Data practices has achieved a more meaningful representation of the same data on the Web. Yet, this does not mean such data is easier for the average stakeholder to locate, access, or most importantly, re-use. Individuals facing these hurdles are typically more acquainted with file formats such as generic JSON, XML, basic CSV or other legacy formats such as XML-based Keyhole Markup Language (KML) or GPS Exchange Format (GPX), therefore finding the sophisticated nature of the RDF format overwhelming. Unfortunately, the emergence of a wide number of tools supporting people to publish their data as Linked (Open) Data[3], has not been complemented by approaches supporting

*Corresponding author. E-mail: attard@iai.uni-bonn.de.
[1]`http://lod-cloud.net/`
[2]`http://lodlaundromat.org/`
[3]`http://www.w3.org/wiki/LinkedData` (Accessed on 21 August 2016)

non-experts to consume existing Linked Data in formats other than RDF [2]. Such tools and approaches would be vital to aid non-experts to exploit Linked Data even though either they are not able to understand and interpret it, or have a system which understands a different format.

We here propose the *ExConQuer Framework*[4] (Explore, Convert, and Query Framework); a set of open source tools[5] whose aim is (i) to facilitate the publication and consumption of RDF data in a wide variety of generic, legacy or domain-specific formats[6], as well as (ii) to enable stakeholders to easily re-use persisted transformations. For these reasons, the ExConQuer Framework is also ideal to introduce Linked Data (and the SPARQL querying language) to new users. The framework is based on the concept of *RDF softening*. In contrast to the semantic *lifting* of data into RDF, which addresses the enrichment, mapping, and transformation of semantically shallow formats, the softening process is then:

> *The generation of domain-specific RDF data views in semantically-shallow representation formalisms.*

This will enable stakeholders to more easily obtain, interpret and re-use existing Linked Data in conventional formats. Moreover, any transformations executed on the data are persisted to enable their re-use. Initiatives such as the one undertaken by the W3C CSV on the Web working group[7], which aims to standardise JSON-LD serialisation, promise to lower the entry barrier to Linked Data re-use. Yet to the best of our knowledge, very few approaches address the need for the provision of semantically-rich RDF data in shallower formats. Although this might appear to be counter-productive, it is favourable to offer the reduction of a degree of semantics in favour of an increase in the *degree of (re)usability* by stakeholders who would otherwise refrain from using the data. Through retaining provenance information we also ensure that the softening process does not result in the loss of the richness of RDF representation, and users are also given the option to lift back the results to RDF.

Based on the motivation of providing stakeholders with a tool that enables them to consume Linked Open Data easily without requiring previous knowledge of RDF, SPARQL, or the datasets' underlying schema, we provide the following contributions as part of the ExConQuer Framework:

- The **Query Builder Tool**[8]: enables users to explore, query, and convert datasources (datasets or subsets) through endpoints;
- **RDF2Any API**: provides the functionality to query and convert RDF datasources into a number of different formats through RDF softening;
- The **ConQuer Ontology**[9]: used to represent transformations carried out in the Query Builder;
- **The Transformation Explorer**[10]: a faceted browser that enables users to explore and re-use Linked Data Publications (all information generated during the use of the Query Builder Tool, such as the query used, the datasource queried, the data formats, etc.);
- **Evaluation**: a usability evaluation on the tools within the ExConQuer Framework, as well as a further effort evaluation that analyses the time and effort required with or without the ExConQuer Framework.

We continue this paper by discussing related work in the literature in Section 2. We provide our approach in Section 3. Then we discuss the led evaluation in Section 4, and provide an overview of where the ExConQuer framework is being used in Section 5. We finally discuss some limitations and future work in Section 6 before providing our concluding remarks in Section 7.

2. Related Work

Our approach is varied in nature, comprising data exploration, query generation, data views, and a provenance-aware management system. To the best of our knowledge, there is no Linked Data consumption framework with all the functions as the one we propose. Yet, there are a number of tools that tackle the different approaches separately.

[4]More information on the framework, including source code and evaluation results, can be found here: `http://eis.iai.uni-bonn.de/Projects/ExConQuer.html`

[5]Source code on Github: `https://github.com/LinDA-tools/QueryBuilder`

[6]While hundreds are in existence: `http://en.wikipedia.org/wiki/List_of_file_formats`, we here focus on the more popular ones such as JSON, CSV and RDB

[7]`http://www.w3.org/2013/csvw/wiki/Main_Page` (Accessed on 21 August 2016)

[8]`http://butterbur22.iai.uni-bonn.de:3000/query/builder`

[9]`http://purl.org/eis/vocab/cqo`

[10]`http://butterbur22.iai.uni-bonn.de/pam/`

2.1. Linked Data Exploration Systems

In the ExConQuer Framework we enable users to explore datasources in order to identify if and how the data they require is represented in existing open datasets. Therefore we here explore various data exploration systems.

In [10], Marchionini distinguishes between *lookup* and *exploratory search* activities. Lookup activities are done to satisfy specific information needs, such as searching for a known item, where the user has defined keywords to use. On the other hand, exploratory search refers to cognitive consuming search tasks, such as learning or investigation. Here, the information need is less well-defined than in a lookup activity and the keywords are not known in advance, therefore also evolving during the activity. In our approach we cater for both activities, where users are given both results that exactly match the specified keyword, and also results that are related to that keyword, as well as being given the option to freely explore the datasource in question by viewing all contained classes and their subclasses.

Tvarozek and Bieliková [20] attempt to facilitate exploratory search by extending their own base browser through the implementation of three search paradigms; keyword-based, view-based, and content-based. The browser also enables dataset exploration through adaptive result overviews and incremental graph-based resource exploration. A drawback for using this approach is the possibility of information overload, since a huge dataset might result in an enormous amount of facets or nodes.

Heim et al. [6] use Facet Graphs in their approach to build semantically unique queries. Users are given the option to choose the result set they need, as well as the facets to filter it. Both are represented as nodes in a graph visualisation and enable them to produce a personalised interface to build search queries. Compared to the previous approach in [20], by enabling users to enter keywords the authors reduce the risk of information overload.

In [1], Araújo et al. present Explorator, a tool for exploring RDF data through direct manipulation. Users are enabled to explore a semi-structured RDF database through browsing and searching. While the led experiments and studies indicated that users with a basic knowledge of RDF were able to use the tool, the authors also point out that the Explorator is better suited to advanced users who have solid knowledge about RDF, further motivating our approach.

Popov et al. [13] propose Visor, a multi-pivot approach that allows users to explore datasets from multiple points in the graph. Visor consists of a generic data explorer tool that can be configured on any SPARQL endpoint. Here, a user is able to explore existing classes in the dataset at hand, the related properties and classes, and individual instances. A graph is then rendered in order to show the user selection and the relations between them (if any). Visor enables users to query a user's selection by creating custom spreadsheets, and then convert them to CSV or JSON.

While numerous tools that enable users to explore Linked Data exist, most of them are targeted for more experienced users who have some knowledge of either RDF or the data's underlying schema. Therefore, such tools are unsuitable to fit our aim of lowering the entry barrier towards re-using Linked Open Data.

2.2. SPARQL Query Builders

The first process towards achieving re-usability is data access. Linked Open Data is usually accessible on data portals or catalogues through SPARQL endpoints or data dumps. The latter method for accessing data has the disadvantage of generally resulting in a large bulk of data, with the user having no control to get specific data (such as a subset) from the data the provider made available as a dump. Moreover, data might also be outdated. While SPARQL endpoints allow thorough control over what data to access, then there is the disadvantage of having to use SPARQL, and using SPARQL to search through data stores is a tedious process and limits data access to Semantic Web practitioners [3,4]. This is mainly due to two reasons; (i) because of the syntax barrier, and (ii) due to the heterogeneity of the data and its schema. As yet, there are few tools that help inexperienced users with respect to the creation and editing of SPARQL queries.

Russell and Smart [15] present NITELIGHT, a tool that enables users to create SPARQL queries using a set of graphical notations and GUI-based editing actions. NITELIGHT uses a visual query language, vSPARQL, to provide graphical formalisms for SPARQL query specification. Users can construct a query through dragging and dropping ontology elements. This approach, while suitable for users with at least a minimal understanding of the SPARQL query language, is not suitable for users who do not know SPARQL or the underlying schema of the dataset to be queried.

Similar to NITELIGHT, Haag et al. [5] also implement a visual approach. The authors define it to be a novel approach for visual SPARQL querying based on the filter/flow model. Thus, no structured text input is required, rather, queries can be generated entirely through the use of graphical elements, and filter restrictions are shown, rather than a representation of the complete query. While this approach does not require knowledge of the SPARQL query language, users are expected to be familiar with the Semantic Web and the filter/flow concepts. Moreover, while this approach allows users to query a dataset, they need to know if and how the information they need is available in the dataset in question.

In contrast to the above, in [14] Pradel et al. present an approach where users can enter a natural language query that is then translated into a formal graph query through the use of query patterns. The aim behind this approach is to hide the complexity of formulating a query expressed in graph query languages such as SPARQL, thus enabling end users to use natural language queries to query ontology-based knowledge bases. The approach described here still has some usability issues. For instance, only English and French can be used as natural languages for the input query. Besides, users who might know the data they need, but not exactly how it is represented in the dataset, will find difficulty in expressing the correct query even if a natural language is used.

QueryMed [17] is the tool that is most similar to our approach for query generation. Focused on the medical domain, this tool enables users with no knowledge of SPARQL to run queries across SPARQL endpoints. The tool requires users to input specific search terms. Users are then given the possibility to filter the results and restrict the query further. A key difference in QueryMed when compared to our approach is that the authors base their search on properties. Thus, when a user selects one or more data stores, the tool displays all the properties within these stores. Apart from resulting in an information overload, this approach is not particularly useful when there many domains involved (e.g. DBpedia), specifically due to the heterogeneity of the data.

2.3. Data Transformations and Exploration Systems

There are a myriad of tools available for converting between data formats, such as Any23[11], Datalift[12] [16], Db2triples[13], and METAmorphoses[14] [19]. However, there are very few tools that enable the conversion of RDF to other, less semantically rich formats (such as [18]). Considering RDF is much more expressive than most other formats, it is understandable that efforts and interest are focused in that direction, however we need to cater for users who require the conversion of Linked Open Data (which is generally available in RDF) to a format they understand which is also compatible to their native systems, such as Microsoft Excel, Open Office, R, or Tableau. Albeit this might result in some loss of information, the advantages outweigh this shortcoming since it will encourage users to exploit such data, rather than being deterred due to unfamiliarity with Linked Data or RDF.

The Transformation Explorer, a provenance-aware management system, is a core contribution within this paper. The aim behind this tool is to provide a means for users to explore and re-use what we call *Linked Data Publications*. A Linked Data Publication consists of all the information generated in the transformation of data, including the SPARQL query used, its description, the datasource(s) queried, the initial and target data formats, and the user generating the Linked Data Publication instance.

In [11], Marie and Gandon survey existing Linked Data based exploration systems, however all the systems they review are based on exploring data, rather than Linked Data Publications which represent the data, as well as the transformations made on it. SPARQLpedia[15] is more similar to what we propose, in that it is a service that allows users to submit SPARQL queries in a searchable repository. The Transformation Explorer follows the same concept, however through retaining provenance information we enable users to not only browse existing queries, but also re-execute them to get updated results or even edit them to refine their query.

[11]`https://any23.apache.org/download.html`
[12]`http://datalift.org/`
[13]`http://www.w3.org/2001/sw/wiki/Db2triples`
[14]`http://metamorphoses.sourceforge.net/`
[15]`http://composing-the-semantic-web.blogspot.nl/2009/01/sparqlpedia-sharing-semantic-web.html`

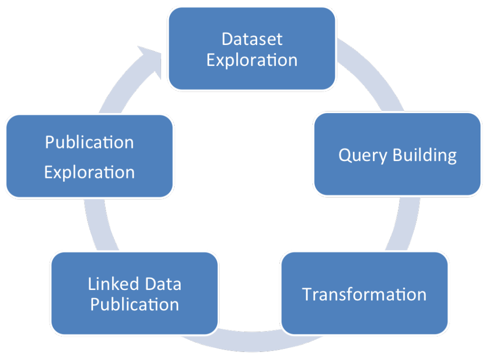


Fig. 1. Abstraction of the ExConQuer Framework Processes

3. Approach

The ExConQuer Framework assists data publishers and consumers in exploiting and re-using Linked Data by providing tools that enable them to easily and simply explore, query, transform, and publish Linked Data. Figure 1 shows an abstract overview of the processes within the framework.

Consider a user who requires to use data on actors from the UK. Through the first stage (*Dataset Exploration*), the user can explore the available datasource, e.g. DBpedia. The user discovers that actors are represented by the class 'Actor'. The user then generates a SPARQL query in the *Query Building* step, adding a filter in order to obtain data only about actors having UK as their nationality, and including information about their age and height. The user then has the option to *Transform* the query results into into various formats. Since the user wants to explore and further re-use the data in Microsoft Excel, he converts the results to CSV. The querying and transformation processes are then represented as a *Linked Data Publication*. Through the Transformation Explorer, the user can *Explore* all previously-generated Linked Data Publications and proceed to re-use, share, or edit them by executing further transformations. Deciding he wants actors over 30 years of age, the user finds the Linked Data Publication generated by his previous query, edits his query by adding a filter, and re-downloads the new results in CSV.

The abstract overview in Figure 1 is implemented through the tools provided within the ExConQuer Framework; namely the the *Query Builder Tool* (Section 3.1, the *RDF2Any API* (Section 3.1.1), the *Transformation Explorer* (Section 3.2), and the *ConQuer Ontology* (Section 3.2.1). Figure 2 shows an overview of the architecture within the framework, and how the

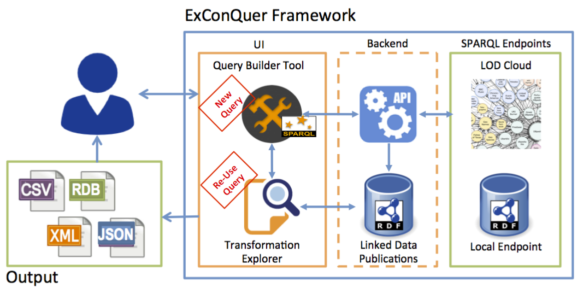


Fig. 2. ExConQuer Framework Architecture

various tools interact with each other. The user can create a SPARQL query through the Query Builder Tool, then query a datastore (through a SPARQL endpoint) through API calls. Once happy with the results, the user can export them in a number of different formats, and re-use them accordingly in his or her native system. Information pertinent to the executed processes is then persisted in a triple store as Linked Data Publications. The latter are represented with the ConQuer Ontology which we propose for recording the provenance information of the transformation. The represented data includes the queried datasource, the SPARQL query, the format conversion, etc. A user can access all this relevant information through the Transformation Explorer, which allows a user to re-use existing resultsets or modify them through the Query Builder.

3.1. *Query Builder Tool*

In the ExConQuer Framework we enable users to explore existing open datasets. We target users who either do not know the content of the datasource in question, or otherwise do not know how specific data is represented in this datasource. Our approach is intended to be particularly user friendly and simple, to allow non-experts to easily use the tool to achieve the goal of re-using open data. An additional advantage of this simplicity is that the tools can be used to introduce Linked Data to new users, as well as helping them to learn the SPARQL query language. Through the RDF2Any RESTful API and by using the datasets' schema, the Query Builder Tool (shown in Figure 3, available online: `butterbur22.iai.uni-bonn.de:3000/query/builder`), enables users to navigate through classes, subclasses, instances, and properties in a somewhat similar manner to a faceted browser, without requiring them to know the structure of RDF data. The API calls concerned with this exploration task are made up of a number of actions that essentially hide the RDF data model and help in the exploration of RDF data and the underlying structure (e.g. to get class labels). This API hence encapsulates the functionality of this tool, and can therefore be re-used in other frameworks.

3.1.1. *Dataset Exploration*

Figure 3 shows different parts of the UI of the Query Builder Tool. The provided exploration functions are particularly useful for users who do not know exactly what data from the available linked datasets is useful for their purpose, or for those who do not know the underlying schema behind the dataset in question.

In Step 1, the user can select any datasource (usually a SPARQL endpoint containing one or more datasets) from the auto-complete drop down list, or otherwise add a new endpoint. In Step 2 the user can then proceed to explore the classes contained in the selected datasource. Here the user can either view all classes, or view the classes which match a given keyword. The user also has the option (by clicking on the plus button) to expand the view and show the subclasses of the selected class, if any are available.

Consider the API call required for the above process. This call abstracts the complexity required to get all classes matching a given keyword. After the user has selected the datasource to explore (Step 1), the user enters a keyword and the API call containing the following SPARQL query is executed:

```
SELECT distinct ?class ?label WHERE {
  { ?class rdf:type owl:Class }
  UNION
  { ?class rdf:type rdfs:Class }.
  OPTIONAL { ?class rdfs:label ?label . }
  FILTER(
    ( bound(?label)  &&  REGEX(?label, "\\b%%
        Search-String%%","i") ) ||
    REGEX(str(?class), "\\b%%Search-String
        %%","i")
    )
} ORDER BY ?class
```

where the *%%Search-String%%* variable will be replaced by the keyword entered by the user. This query is used to search within the selected datasource to look for resources of type *owl:Class* or *rdfs:Class*. Here the user also has a choice (through the UI) to search for the latter resources either through just the resources' labels, or otherwise extend the search to also include resources' URIs. In the case of the former, the query would be a little different, in that both the OPTIONAL clause and the REGEX component for the class would be removed. The REGEX component of the query will enable the classes to be searched and returned to the user on the fly in an autocomplete manner.

In Step 2, along with the classes and subclasses, a number of example instances are displayed, ordered by the amount of local backlinks each instance has. The SPARQL query used to obtain this data is as follows:

```
SELECT ?label ?instance {
  ?instance rdfs:label ?label .
  {
    SELECT DISTINCT ?instance (COUNT(?x) AS ?
        cnt) WHERE {
      ?instance a <%%Concept-URI%%> .
      ?x ?p ?instance .
    }
    GROUP BY ?instance
    ORDER BY DESC(?cnt)
    LIMIT %%limit%%
  }
  FILTER(langMatches(lang(?label), "EN"))
}
```

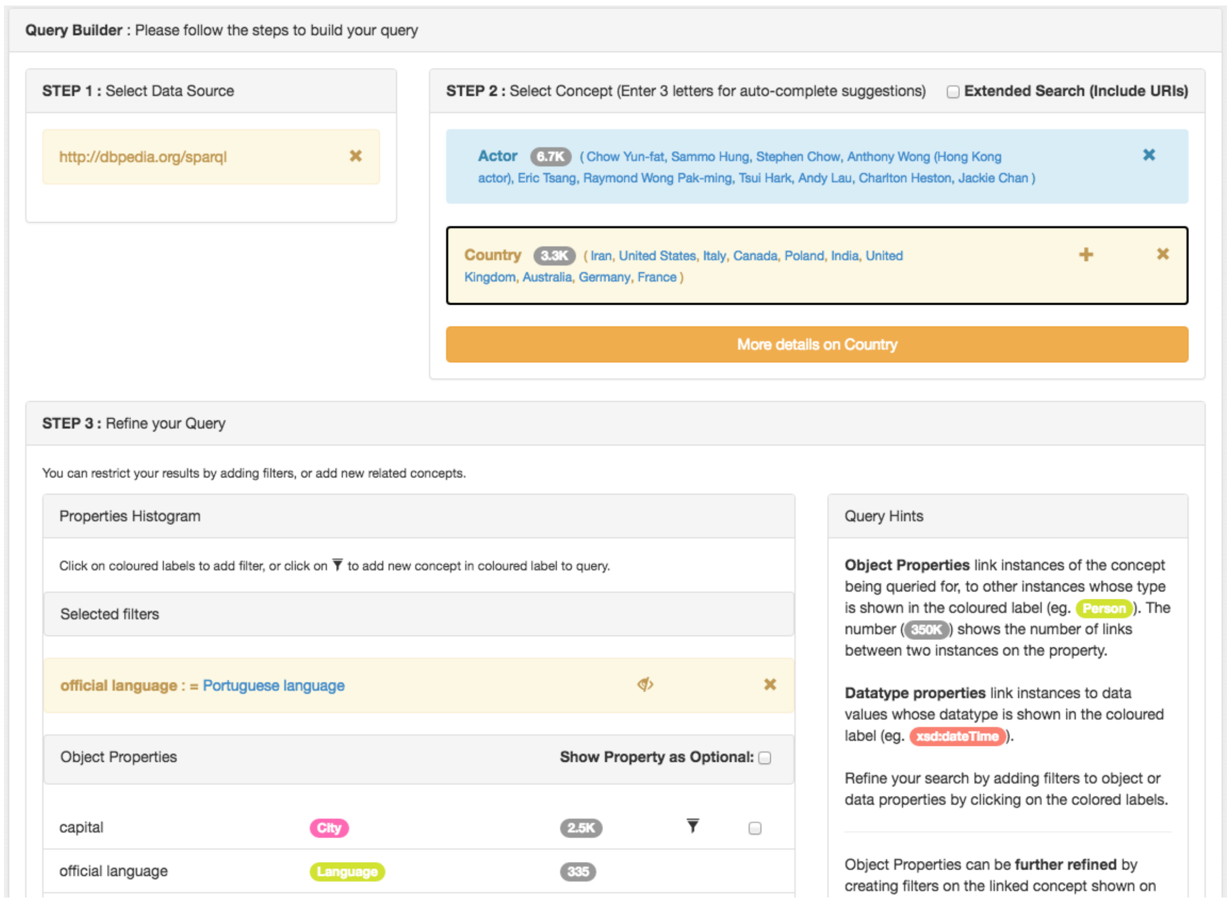
Once a class is selected, the user can proceed to Step 3; the Properties Histogram view, where all the properties of the selected class are shown. The view is divided into *Object Properties* and *Data Type Properties*. The former is when a property is defined as an *owl:ObjectProperty* and thus expects an object URI resource as its range, whilst the latter is for properties defined as an *owl:DatatypeProperty*, and hence are expecting a data literal as their range. The following listing shows the query used to retrieve both Datatype and Object properties of the selected class, as well as for its superclass.

```
SELECT DISTINCT ?property ?label WHERE {
  ?property rdf:type owl:%%Type%%.
  ?property rdfs:label ?label .
  { ?property rdfs:domain <%%Concept-URI%%> .
      }
  UNION { ?property rdfs:domain ?superClass .
      }
  {
    SELECT DISTINCT ?superClass { <%%Concept-
        URI%%> rdfs:subClassOf* ?superClass .
         }
  }
    FILTER(langMatches(lang(?label), 'EN'))
}
GROUP BY ?property ?label
```

In Step 3, the user can add filters to restrict the results as well as add optionals to the end results. Furthermore, users can refine their query by adding other related classes to the query (multiple class query).

3.1.2. *Query Generation*

Apart from enabling users to explore datasources, the main task of the Query Builder tool is to aid users to generate a SELECT SPARQL query, without re-



Query Builder : Please follow the steps to build your query

STEP 1 : Select Data Source

http://dbpedia.org/sparql ✖

STEP 2 : Select Concept (Enter 3 letters for auto-complete suggestions) ☐ **Extended Search (Include URIs)**

Actor 6.7K (Chow Yun-fat, Sammo Hung, Stephen Chow, Anthony Wong (Hong Kong actor), Eric Tsang, Raymond Wong Pak-ming, Tsui Hark, Andy Lau, Charlton Heston, Jackie Chan) ✖

Country 3.3K (Iran, United States, Italy, Canada, Poland, India, United Kingdom, Australia, Germany, France) ✚ ✖

More details on Country

STEP 3 : Refine your Query

You can restrict your results by adding filters, or add new related concepts.

Properties Histogram

Click on coloured labels to add filter, or click on ⏷ to add new concept in coloured label to query.

Selected filters

official language : = Portuguese language ✖

Object Properties **Show Property as Optional:** ☐

capital	City	2.5K	⏷	☐
official language	Language	335		

Query Hints

Object Properties link instances of the concept being queried for, to other instances whose type is shown in the coloured label (eg. Person). The number (350K) shows the number of links between two instances on the property.

Datatype properties link instances to data values whose datatype is shown in the coloured label (eg. xsd:dateTime).

Refine your search by adding filters to object or data properties by clicking on the colored labels.

Object Properties can be **further refined** by creating filters on the linked concept shown on

Fig. 3. Query Builder Tool

quiring prior knowledge of SPARQL or the datasets' underlying schema. This tool enables users to generate a SPARQL query through a user-friendly interface equipped with auto-complete features. Similar to the exploration function of the Query Builder, the query building function is also enabled through the consumption of the RDF2Any API, where the user selection for the classes and properties is converted into a SPARQL query that is then executed on the selected datasource.

In order to generate a SPARQL query, the user can follow exactly the same procedure as explained in Section 3.1.1. After selecting the datasource and class to query, the user can proceed to select the properties to be included in the resultset. Properties can be freely selected to be included or excluded from the results, and a click on a property allows the user to define a filter. Additionally, at this stage the user can also select to add the object type class of the relevant property as a new concept, hence obtaining a *multiple class query* which enables the user to add further filters on the the selected classes. An example of a multiple class query is when the user would like to get as results any actors born after 1900 whose nationality is a country where the official language is Portuguese (as shown in Figure 3). This is done by selecting Actor as the first class and adding a filter on the birth year. Then the class of the actor's nationality (Country class) is added as a second class. Finally a filter on the official language property is set to only return countries where the official language is Portuguese. Throughout the query building process, the query, which is generated on the fly, is displayed. Thus, once the user has made the preferred selections, the generated query is previewed and can be edited if this is required. Finally, the user has the option to first preview a subset of the results, and then proceed to export the full result set.

3.1.3. *Data Transformation*

Provided within the Query Builder Tool, the Transformation function is aimed towards users who need the resultset in a format different than RDF. This might

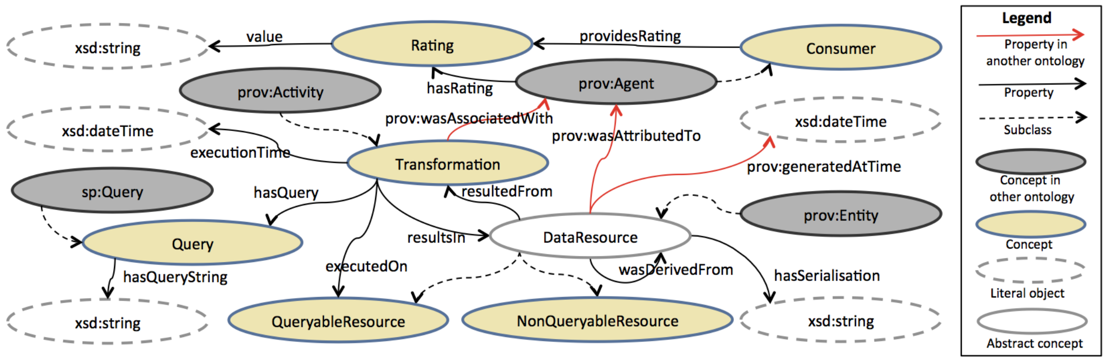


Fig. 4. ConQuer Ontology for Modelling Linked Data Publications

be because their native system understands other formats, or simply because they find results in another format more easily readable and interpretable. This *softening* does indeed result in a certain degree of loss in semantics. Yet, this is compensated through retaining links with the original RDF data and other relevant information through the ConQuer Ontology (see Section 3.2.1). This means that it is always possible to obtain the original data in RDF through exploiting the provenance information recorded for every transformation and resultset.

The transformation process consists in converting the results in RDF to a number of different formats through the consumption of the RDF2Any API. Currently, the conversions provided are from RDF to CSV, JSON, and RDB, as well as a more advanced configurable conversion. The latter allows a user to convert RDF into potentially any output format, such as XML, KML, TSV (tab separated values), etc. The exception are formats which require memory storage, such as RDBMS serialisation, which require the storing of foreign key values. The use of the Generic Conversion requires some knowledge about the dataset(s) to be converted, and the user is required to pass required parameters through a template. Apart from being easily extendible with further converters, the transformation process provides the additional advantage that a user can directly convert the required subset of the datasource in question, rather than converting a bulky data dump. We manually validated the correctness of the various conversions for various queries on different datasources. While we confirm there is a loss from the rich representation of RDF, the essence of the data is retained and the provenance information allows us to retain the link to the original data and the transformations for reproducibility.

3.2. *Transformation Explorer*

All the processes executed through the ExConQuer Framework generate what we call a Linked Data Publication, which is basically what users can share, re-use, explore, and edit. Thus, a Linked Data Publication consists of all the generated information, including the SPARQL query used, its description, the datasource(s) queried, the initial and target data formats, and the user generating the Linked Data Publication instance[16]. We represent all this data using the *ConQuer Ontology*, shown in Figure 4. All generated Linked Data Publications can then be explored using the *Transformation Explorer* (available online: `http://butterbur22.iai.uni-bonn.de/pam/`), which furthermore enables users to re-execute or edit existing queries.

The main aim of the Transformation Explorer is to provide stakeholders with the potential to explore all existing queries and transformations executed on different datasources. In this way, a user is given the opportunity to find any results that match the given requirements. Moreover, if the results are not exactly as a user requires, for example if they are in a different format, or the resulting data is too generic/specific, the user can proceed to edit or update the results with minimal effort, through re-loading the Linked Data Publication on the Query Builder Tool.

3.2.1. *ConQuer Ontology*

The ConQuer ontology, through the represented information, not only allows us to represent all possible transformations on an entity through querying

[16]This is not implemented in the online demo as yet, since we wanted to avoid forcing users to register and log in, in order to use the tool.

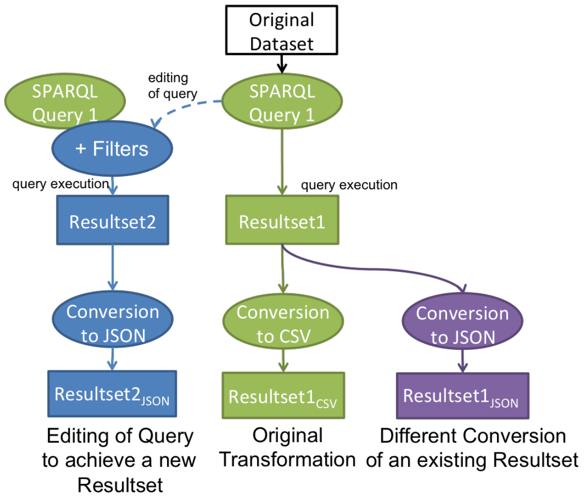


Fig. 5. Example of possible Linked Data re-use scenarios enabled by the ExConQuer Framework and the underlying provenance-aware ConQuer Ontology

and converting, but it also allows us to replicate the resulting Linked Data Publications and edit them to achieve different results. Figure 5 shows how, starting from a transformation on a specific datasource (Original Transformation), a user can re-use the query but execute a different conversion on the resultset, or otherwise edit the original SPARQL query in order to obtain different (more generic, more specific, or otherwise) results. Thus, using the ConQuer ontology to represent our transformations allows us to *soften* RDF into semantically shallower formats without actually compromising on the the richness of RDF representation, as any resultsets in formats other than RDF are linked back to the original data in RDF. Additionally, through the provenance information, we can track the changes to each entity, and also assign a reputation or a rating for the different agents generating the Linked Data Publications.

The main concepts in the ontology are the following:

- **Transformation**: A Transformation represents all the information required to achieve a Linked Data Publication, as described above.
- **Query**: A Query represents a set of statements forming a SPARQL query.
- **Data Resource**: A Data Resource is used to represent a data store. This can be anything from a linked open dataset with a SPARQL endpoint such as DBpedia, to a database or a CSV document.
- **Agent**: An Agent is any entity, whether machine or human, that has some sort of control or authority over the generation of a Transformation instance.

To describe the ontology in an informal manner, a *Transformation* has a *Query* that is executed on one or more instances of a *DataResource* (enabling the representation of federated queries). The latter must be a *QueryableResource*, or, in other words, it should be expressed in one of the serialisations of the RDF data format (RDF/XML, NTriple, Turtle, etc.). The resulting *DataResource*, on the other hand, can be either a *QueryableResource* or a *NonQueryableResource* (formats such as CSV, PDF, etc). Finally, each *Transformation* and *DataResource* are linked through the relevant properties.

Since the ConQuer Ontology is representative of Transformations, thus making the latter class the main concept within the ontology, we define a *Transformation* T as follows:

Definition 1 $T = \{q, d, f_d, r, f_r, a, t\}$

where q is a *Query*, d and r are *DataResource* instances (original resource(s) and resultset), f_d and f_r are the serialisation formats of d and r respectively, a is an *Agent*, and t is the time the transformation was executed. Hence, a generates T, which represents a Linked Data Publication instance. The latter results from applying q to d and then obtaining the final Linked Data Publication by converting f_d to f_r. This means that $r \subseteq d$, as the user can query to get all, or part of resource d.

In the ConQuer Ontology we re-use concepts from the SPIN vocabulary [7], which is used to represent re-usable SPARQL queries as templates, and also from the PROV-O ontology [8], used to represent provenance information. The use of SPIN to represent SPARQL queries not only enables the direct querying of the queries themselves, but also allows the represented knowledge to be re-used in any frameworks or tools using the SPIN vocabulary. The re-used concepts are:

- **sp:Query**: A SPIN concept which represents a SPARQL query. This concept enables us to search within the persisted *Query* instances.
- **prov:Activity**: A PROV-O concept representing something that occurs over a period of time and either interacts with or acts upon *prov:Entity* instances. *prov:Activity* instances can include transforming, consuming, using, or generating entities.

– **prov:Entity**: An *Entity* can be physical, digital, conceptual, or any other thing with a fixed set of aspects.
– **prov:Agent**: This concept represents something or someone who bears some sort of responsibility for an *Activity* taking place or for the existence of an *Entity*.

3.2.2. Linked Data Publication Exploration and Management

We implemented the *Transformation Explorer* as a management tool that enables the exploration of Linked Data Publications with the aim of encouraging their re-use. The motivation behind providing such a tool is that queries are re-usable, and a single query might be the answer to many users' requirements. Besides, the Transformation Explorer also enables users to persist and re-use complex SPARQL queries. The re-use of queries is particularly useful when a dataset is frequently updated, as a user can simply re-run the query in question to get the updated results. We query the persisted instances of the Linked Data Publications and publish them through a faceted browser (Exhibit[17]). Through the use of the ConQuer ontology, the Linked Data Publications have queryable metadata that enables users to search for specific instances using various criteria, such as by the datasources used and the classes queried for. Moreover, a user would be able to search by Agent if the user is required to log in before using the Query Builder Tool[18]. Through the persistence of such provenance information, users could query Linked Data Publications according to Agents who have the reputation of providing the best data for the intended use. This tool thus allows users to share, explore, and directly edit (through the Query Builder or otherwise) and re-use Linked Data Publications, whilst keeping data lineage intact.

4. Evaluation

The purpose of this section is to discuss the evaluation led on the ExConQuer Framework. Our framework is intended for the use of stakeholders who are not familiar with RDF or the Linked Data paradigm. This does not exclude stakeholders who already use Linked Data in one way or another, simply because users who are not familiar with the underlying complexity are unable to exploit Linked Data to its fullest potential. For example, a user downloading a data dump of a linked dataset is hardly exploiting the potential of the data in question. For these reasons, the framework requires to be very user-friendly and provide simple access to the required functionality, whilst also abstracting the underlying complexity. This evaluation is divided in two parts as follows:

1. A comprehensive survey intended to assess the usability of the tools;
2. A shorter survey concerned with analysing the time and effort required to re-use open data with and without the ExConQuer Framework.

For each part we use a different set of evaluators. The used surveys and complete results are available online[19].

4.1. Usability Evaluation

Since the aim of this evaluation is to identify whether the ExConQuer framework helps or encourages people in the re-use of Linked Data, we shared this evaluation with relevant partners or colleagues who, to some extent or another, had contact with Linked Data. In total we had 27 evaluators, who, considering research such as Nielsen's [12], should be able to point out even more than the most relevant usability issues in the evaluated tools. Their domains differ in nature (such as education, healthcare, research, consulting, industry and marketing). 6 of them do not use Linked Data at all, whilst 21 use it either personally, in their work, or both (mostly for analysis, visualisation and integration). All the evaluators specified more or less the same processes while interacting with Linked Data, namely searching for existing data, accessing and gathering it, cleaning it, integrating it, leading out analyses, and consuming it by visualising it or in other ways such as data mashups. Apart from other issues, nearly all evaluators pointed out that the format of the data hindered them from re-using it, and very commonly data is also incomplete or invalid. Moreover, data might not be accessible at all. 11 out of the 27 evaluators are not familiar with the SPARQL query language, so we were able to interpret the results considering the two different target users.

For this usability evaluation we constructed a survey consisting of 25 questions and split it into three

[17]`http://www.simile-widgets.org/exhibit/`
[18]This is not currently implemented in our online demo.
[19]`http://eis.iai.uni-bonn.de/Projects/ExConQuer.html`

		Strongly Agree		**Agree**		**Neither Agree nor Disagree**		**Disagree**		**Strongly Disagree**	
		LD Users	Non Users	LD Users	Non Users	LD Users	Non Users	LD Users	Non Users	LD Users	Non Users
	Query Builder										
1.	Do you agree that it was easy to execute this task?	6	1	9	6	2	2	1	0	0	0
2.	Do you agree that this tool would be useful in your SME/Company to access, explore and query open datasets?	4	2	11	5	2	2	1	0	0	0
	Transformation Explorer										
3.	Do you agree that it was easy to execute this task?	3	1	9	5	2	2	2	1	2	0
4.	Do you agree that this tool would be useful in your SME/Company to re-use saved data access queries?	6	1	6	6	3	2	2	0	1	0

Table 1

Results for Tool Evaluation (Complete results: `http://eis.iai.uni-bonn.de/Projects/ExConQuer.html`)

sections, namely questions on the current means and methods of accessing and using Linked Open Data (if any), questions on the Query Builder Tool, and finally questions on the Transformation Explorer. Where relevant, we used the Likert scale [9] to assess the evaluators' perception of the tools.

4.1.1. *Query Builder Tool Evaluation*

In order to have a better insight, the evaluators were asked to describe their current process of querying Linked Data. 6 of the evaluators directly specified they use SPARQL to query Linked Data. The rest either do not use Linked Data (6), or use other methods for querying such as running test queries, using query designers, or exporting to Microsoft Excel (15). The evaluators were then asked to access the Query Builder Tool, explore a dataset, formulate a SPARQL query including filters, and download and convert the results in the preferred format. Questions 1 and 2 in Table 1 show the results for the evaluators' impression of the Query Builder tool. The evaluator who replied with 'disagree' in both question 1 and 2 was of the opinion that tutorials or demo videos would have been helpful with executing the given task. For question 2 in Table 1, almost all the evaluators (22) agreed that they would find this tool useful (to some degree or another) in their SME/Company/Academic Entity. When asked if the Query Builder is a better approach than their current way of consuming Linked Data (question is available in complete survey online), only 5 replied 'not sure' while the others all agreed that it would be better.

From this part of the evaluation we can conclude that while this tool still requires some improvements with regard to usability, it is however generally deemed to be useful by the target stakeholders (both experts/non-experts, and users/non-users of Linked Data) and is an improvement on their current methods of exploiting Linked Data (if any).

4.1.2. *Transformation Explorer Evaluation*

For this part of the evaluation, the evaluators were asked to use the Transformation Explorer to search for the Linked Data Publication they just created in the previous section of the survey, then re-load and edit it on the Query Builder Tool. The users were able to use a number of facets to filter the results. For this tool, the responses to question 3 in Table 1 were somewhat varied, however the majority of the evaluators still agreed that the tool is quite easy to use, and that it would be useful to their company. Most of the comments from the negative replies pointed out that the tool took quite long to load, and one evaluator who selected 'strongly disagree' commented that we show too many details (such as the SPARQL query). On the other hand, the other evaluator who selected 'strongly disagree' for question 3 still thought that the tool would be very useful in his context. When asked question 4, the evaluators' replies were mostly positive. Yet again, this indicates that while the tool needs improvement, mostly efficiency-wise, the majority of the evaluators still consider the tool to be useful.

4.2. *Effort Evaluation*

In this evaluation we required to analyse if the ExConQuer framework makes the open data re-use process more easy or efficient for the users. This evaluation consisted in asking the evaluators (different from the evaluators in the usability evaluation) to execute a simple task requiring obtaining some data from DBpedia, with and without the ExConQuer tools. In total we had 20 evaluators who, similar to the previous evaluation, have some contact with Linked Data but do not necessarily know SPARQL, RDF, or the datasets' underlying schema. In fact 9 of the evaluators stated they did not use SPARQL queries on a frequent basis, and two of whom did not even know anything about the querying language.

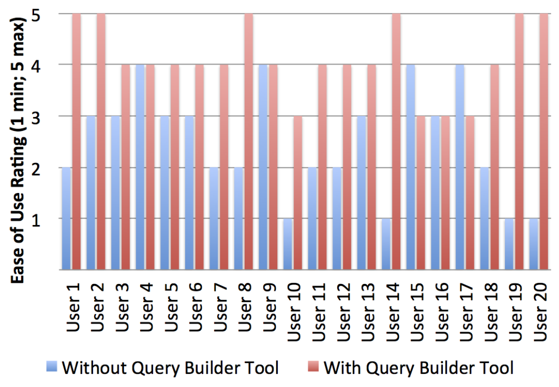


Fig. 6. Comparison of ease of use rating for executing the task, with and without the Query Builder Tool (where 1 is not easy, 5 is very easy)

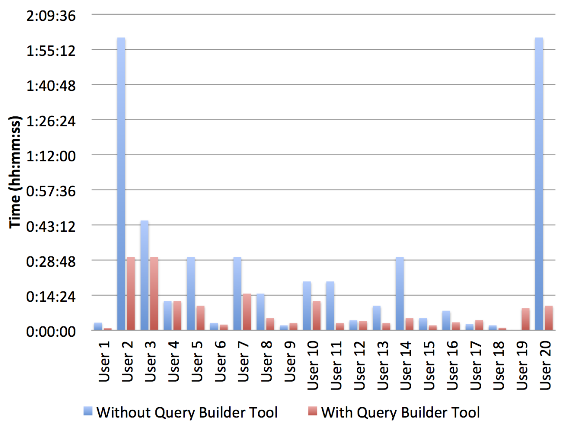


Fig. 7. Comparison of time taken to execute the task, with and without the Query Builder Tool

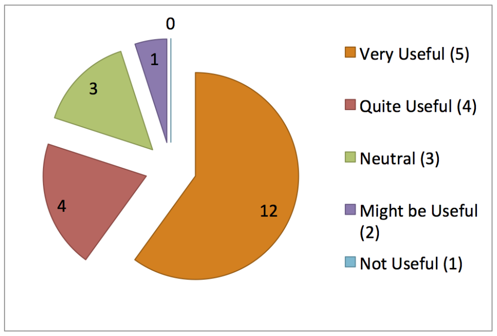


Fig. 8. Results for rating whether the Query Builder Tool is useful to learn SPARQL

In order to determine whether the Query Builder improved on the time and effort required to obtain open data, we defined a simple task which required the users to get some data from DBpedia as follows:

Task: Get all actors whose nationality is a country where the national language is English.

The evaluators were thus required to execute this task using their usual method for accessing open data. In this evaluation, all the users attempted to use the DBpedia SPARQL endpoint, albeit showing different levels of ease and efficiency. Then, the users were required to get the same data using the Query Builder tool. As part of the evaluation, the users entered the ease with which they managed to execute the task as well as the time taken to do both tasks (separately). The results are shown in Figures 6 and 7.

For the task using the usual method for accessing open data, the users who were not so familiar with SPARQL ended up using a search engine to obtain the required SPARQL query. Figure 6 shows the ratings given (for both methods) to the effort required to do the given task, where 1 means 'not easy' and 5 means 'very easy'. For the users' usual method, 4 users rated the difficulty to be quite easy (rating = 4), the results of all the other 16 users ranged from neutral (rating = 3) to not easy (rating = 1). On the other hand, using the Query Builder Tool, 4 users rated the ease-of-use of the tool to be neutral (rating = 3), 10 rated it to be quite easy (rating = 4), whilst 6 rated the tool to be very easy to use (rating = 5).

With regard to the time taken, as shown in Table 7, only 1 evaluator (User 4) took the same time in both methods, finding both approaches equally easy to execute, whilst 2 evaluators (Users 9 and 17) took more time using the Query Builder, where one rated the ease of use to be equal, and the other said it is less easy to use the Query Builder. All the rest of the evaluators took less time to execute the task using the Query Builder Tool as opposed to using their usual method, namely 12 minutes and 1 second less on average (Table 2). One user (User 19) did not even manage to execute the given task at all, having no idea how to access the DBpedia datasource. Two users (Users 2 and 20) took a particularly long time in executing the task using their preferred method; about 2 hours each. Given that the users specified that they are not very familiar with SPARQL queries, we can safely assume that it is quite reasonable that they took two hours to do the task. First they required to figure out how to do a SPARQL query, which has quite a steep learning curve. Possibly, they did this through learning by example, since they only needed to do one task in this case. Then they needed to understand the DBpedia schema to identify how the re-

	Without Query Builder	With Query Builder
Average time	0:24:11	0:08:13
Maximum Time	2:00:00	0:30:00
Minimum Time	0:02:00	0:00:52

Table 2

Average, Maximum, and Minimum time taken to execute the task, with and without the Query Builder Tool.

quired concepts are represented, before finally producing the SPARQL query which provides the required results. Being unexperienced in SPARQL, it is most probable the users needed to do various corrections to the query before managing to obtain the correct one.

Taking into consideration the results of the effort evaluation, we can conclude that the tool enables users to more easily and more efficiently execute a data gathering task from a datasource with a SPARQL endpoint. Whilst it is not as useful for users who are very familiar with SPARQL queries, the Query Builder Tool was considered to be quite useful to introduce and teach SPARQL to users who are not familiar with the querying language (see Figure 8), therefore reaching the aim of lowering existing barriers to re-using Linked Data.

5. ExConQuer in Use

The ExConQuer Framework, created as part of the LinDA Project[20], was used by a number of SMEs who participated within the project consortium, as pilot partners or otherwise. The ExConQuer tools were used in the following scenarios, using datasets that vary between open data, government data, and private data.

- A Business Intelligence scenario at Critical Publics[21], an SME headquartered in London that implements strategies to manage relationships with important stakeholders;
- A Water Management scenario, run by Hyperborea[22], an Italian Company specialising in ICT solutions for the environmental management sector;
- A Media Industry pilot, at an Italian broadcaster, TTNEWS24[23];
- A Media Industry pilot led by Piksel[24] (Italian Branch), a company specialised in providing holistic solutions for management of post production scripts and providing advanced media analytics.

Along with other LinDA tools, the ExConQuer framework is also being endorsed in a number of other initiatives, projects, or SMEs, including but not limited to the following. Other collaborations are listed on the LinDA website[25].

- ODINE[26]: An open data incubator where more than 500 SMEs have applied to date;
- Your Data Stories[27]: A project that deals with finding, analysing, and visualising open data;
- Infamous Labs[28]: A software development company that provides high quality technical services related to Smart TVs;
- Open Aire[29]: A large-scale initiative that aims to promote open scholarship and improve the discoverability and re-usability of research publications and data;
- Suite5[30]: An SME working on transforming data streams from multiple sources to analytics and intelligence;
- Weather ex Machina[31]: An SME providing a weather forecasting service based on data aggregation.

Apart from the above initiatives, the ExConQuer Framework is also being exploited directly on DBpedia[32] as a query builder tool and SPARQL query interface.

6. Limitations and Future Work

In this section we cover a number of known limitations of the tools within our framework, and provide possible solutions as future work.

- **SPARQL queries**: Currently the tool limits the creation of SPARQL queries to SELECT queries. The extension of this functionality to include the other types of SPARQL queries, such as ASK and CONSTRUCT queries, would provide users with

[20]`http://linda-project.eu/`
[21]`http://www.criticalpublics.com/`
[22]`http://www.hyperborea.com/`
[23]`https://www.facebook.com/ttnews24/`
[24]`http://www.piksel.com/`

[25]`http://linda-project.eu/linked-projects/`
[26]`https://opendataincubator.eu/`
[27]`http://yourdatastories.eu/`
[28]`http://www.infamouslabs.net/`
[29]`https://www.openaire.eu/`
[30]`http://www.suite5.uk/`
[31]`http://weatherxm.com/`
[32]`http://wiki.dbpedia.org/projects/exconquer`

further control over the Linked Data they are interested in and more flexibility in the end results.

- **Filters**: This version of the Query Builder provides an auto-complete feature during the definition of filters on existing *owl:ObjectProperty*. This helps the user to only enter a filter value that will yield some results. However, sometimes the user is unaware about how the property value is represented in the dataset. For example, a date could be represented in different formats, such as dd/mm/yyyy or mm/dd/yy. The provision of some examples extracted from instances would aid the user to identify the correct representation. In addition to an auto-complete feature, these examples will make it much easier for the user to create a query that includes filters which will ultimately yield the desired results.
- **Schema**: The tool requires the explicit definition of classes and properties through the *owl:Class*, *rdfs:Class*, *owl:DatatypeProperty*, and *owl:ObjectProperty*. Moreover, if the properties do not have a domain and range, then they will not be previewed by the tool since we cannot identify if there is a relationship between such a property and any classes. Currently, these definitions should also be accessible through the datasource's SPARQL endpoint in order for the tool to correctly preview the contents of the dataset(s). These limitations can be solved if the tool identifies classes and properties based on the actual instances in the datasource, rather than through its schema, although this might result in a less-efficient tool.
- **Dependency on endpoint**: The Query Builder tool depends on the availability of the used SPARQL endpoint to function. If the endpoint is down the tool is not able to preview the data. A possible solution to this limitation is to have a local copy of the endpoint, however this is usually not feasible due to the large size of the datasource.
- **Versioning endpoints**: Some use cases might require that the results of a query are reflective of a specific snapshot of a datasource. Storing the results of the original execution of the query would however be unfeasible due to the resulting size of all the queries executed through the Query Builder. A more feasible solution would be for the data producers to implement some sort of versioning for their endpoint. The ExConQuer framework could then be updated to enable users to select on which version of the endpoint they would like the query to be executed on.

7. Conclusion

It is evident that the use of Linked Data principles is increasing at a fast rate, as indicated through the exponential growth of the Linked Open Data Cloud. This increase is also reflected in tools aiding users in the publishing process, yet, tools aiding users to consume and re-use Linked Data are still not that prevalent. We hence identify a niche with regard to approaches that abstract the complexity beneath exploiting linked datasets and propose the *ExConQuer Framework*. In order to provide more simple and workable views of the data, in this framework we transform RDF data into a number of different formats, whilst still preserving the semantic richness of the RDF data model. While during this process we do lose some of the richness of RDF representation, we compromise by preserving the link with the original RDF data through the ConQuer ontology, and still retain the full semantic richness through provenance information. As is evident through the evaluation we performed, the ExConQuer Framework is particularly useful to encourage the re-use of Linked Data by stakeholders who are not familiar with RDF, and are more acquainted with formats such as JSON or CSV. Our framework is also useful for more expert users who are however not able to exploit Linked Data to its full potential due to not being familiar with RDF, SPARQL or the data's underlying schema.

References

[1] Samur Araújo and Daniel Schwabe. Explorator: A tool for exploring RDF data through direct manipulation. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Kingsley Idehen, editors, *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009.*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[2] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[3] Stéphane Campinas. Live SPARQL auto-completion. In *Proceedings of the 2014 International Conference on Posters & Demonstrations Track - Volume 1272*, ISWC-PD'14, pages 477–480, Aachen, Germany, 2014. CEUR-WS.org.

[4] Lin Clark. SPARQL Views: A visual SPARQL query builder for drupal. In *Proceedings of the 2010 International Conference on Posters & Demonstrations Track - Volume 658*, ISWC-

PD'10, pages 161–164, Aachen, Germany, 2010. CEUR-WS.org.

[5] Florian Haag, Steffen Lohmann, Steffen Bold, and Thomas Ertl. Visual SPARQL querying based on extended filter/flow graphs. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, AVI '14, pages 305–312, New York, NY, USA, 2014. ACM.

[6] Philipp Heim, Thomas Ertl, and Jürgen Ziegler. Facet graphs: Complex semantic querying made easy. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010)*, volume 6088 of *LNCS*, pages 288–302, Berlin/Heidelberg, 2010. Springer.

[7] Holger Knublauch, James A. Hendler, and Kingsley Idehen. *SPIN – Overview and Motivation*. W3C, 2011. Available from `http://www.w3.org/Submission/spin-overview/`.

[8] Timothy Lebo, Satya Sahoo, and Deborah McGuinness, editors. *PROV-O: The PROV Ontology*. W3C, 2012. Available from `http://www.w3.org/TR/prov-o/`.

[9] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22:1–55, 1932.

[10] Gary Marchionini. Exploratory search: From finding to understanding. *Commun. ACM*, 49(4):41–46, April 2006.

[11] Nicolas Marie and Fabien Gandon. Survey of linked data based exploration systems. In *Proceedings of the 3rd International Conference on Intelligent Exploration of Semantic Data - Volume 1279*, IESD'14, pages 66–77, Aachen, Germany, 2014. CEUR-WS.org.

[12] Jacob Nielsen. Why you only need to test with 5 users, March 2000. Available from `http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`, accessed 2016-04-28.

[13] Igor O. Popov, M. C. Schraefel, Wendy Hall, and Nigel Shadbolt. Connecting the dots: A multi-pivot approach to data exploration. In *Proceedings of the 10th International Conference on The Semantic Web - Volume Part I*, ISWC'11, pages 553–568, Berlin, Heidelberg, 2011. Springer-Verlag.

[14] Camille Pradel, Ollivier Haemmerlé, and Nathalie Hernandez. Natural language query interpretation into SPARQL using patterns. In *Proceedings of the Fourth International Conference on Consuming Linked Data - Volume 1034*, COLD'13, pages 13–24, Aachen, Germany, 2013. CEUR-WS.org.

[15] Alistair Russell. Nitelight: A graphical editor for SPARQL queries. In *Proceedings of the 2007 International Conference on Posters and Demonstrations - Volume 401*, ISWC-PD'08, pages 110–111, Aachen, Germany, 2008. CEUR-WS.org.

[16] François Scharffe, Ghislain Atemezing, Raphaël Troncy, Fabien Gandon, Serena Villata, Bénédicte Bucher, Fayçal Hamdi, Laurent Bihanic, Gabriel Képéklian, Franck Cotton, Jérôme Euzenat, Zhengjie Fan, Pierre-Yves Vandenbussche, and Bernard Vatant. Enabling linked data publication with the Datalift platform. In *Proc. AAAI workshop on Semantic cities*, Toronto, Canada, July 2012.

[17] Oshani Seneviratne. QueryMed: An intuitive SPARQL query builder for biomedical rdf data abstract.

[18] Alex Stolz, Bene Rodriguez-Castro, and Martin Hepp. RDF translator: A restful multi-format data converter for the semantic web. *CoRR*, abs/1312.4704, 2013. Available from `http://arxiv.org/abs/1312.4704`.

[19] Martin Svihla and Ivan Jelinek. Benchmarking RDF production tools. In *Proceedings of the 18th International Conference on Database and Expert Systems Applications*, DEXA'07, pages 700–709, Berlin, Heidelberg, 2007. Springer-Verlag.

[20] Michal Tvarozek and Mária Bieliková. Generating Exploratory Search Interfaces for the Semantic Web. In *Human-Computer Interaction - Second {IFIP} {TC} 13 Symposium, {HCIS} 2010*, pages 175–186, Berlin, Heidelberg, 2010. Springer.

Acknowledgements The work presented in this article was performed in the EU FP7 LinDA Project[33].

[33] `http://linda-project.eu/`