CORNER: A Completeness Reasoner for SPARQL Queries over RDF Data Sources

Fariz Darari, Radityo Eko Prasojo, and Werner Nutt

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
`{fariz.darari,radityoeko.prasojo}@stud-inf.unibz.it,nutt@inf.unibz.it`

Abstract. With the increased availability of data on the Semantic Web, the question whether data sources offer data of appropriate quality for a given purpose becomes an issue. With CORNER, we specifically address the data quality aspect of completeness. We demonstrate a formal way to state for which topics an RDF source is complete and how to use such statements to automatically analyze whether a given query will return a complete answer over sources. CORNER supports SPARQL BGP queries and can take RDFS ontologies into account in its analysis. If a query can only be answered completely by a combination of sources, CORNER rewrites the original query into one with SPARQL `SERVICE` calls, which assigns each query part to a suitable source, and executes it over those sources. CORNER builds upon previous work by Darari et al. [1] and is implemented using standard Semantic Web frameworks.

Keywords: Data quality, data completeness, query completeness, SPARQL

1 Introduction

In recent years, large amounts of data have been made available on the Semantic Web, which can be accessed by posing queries to SPARQL endpoints. As more data become available, quality of data becomes an issue since data in different sources may be suitable for different usages. In particular, *data completeness* may vary among data sources. Consequently, users who pose a query to different sources may get answers with different degrees of completeness. The question is how to support users in choosing sources over which their queries can retrieve complete answers.

For relational databases, Levy [2] proposed a format for statements about data completeness and studied how to assess the completeness of a query in the presence of such statements. Razniewski and Nutt [3] introduced a general reasoning technique for this problem and provided a comprehensive complexity analysis. Darari et al. [1] developed a framework for completeness reasoning techniques on the Semantic Web. The framework enables one to provide descriptions as to which parts of a data source are complete, called *completeness statements*, and to perform checks whether a given query over such a data source returns a complete result, called *query completeness* checks. The framework supports basic graph pattern (BGP) queries [4] and can take into account RDFS ontologies featuring `subclass`, `subproperty`, `domain` and `range`. Moreover, if a query can be ensured to be complete over a combination of data sources, the framework tells one how to produce a federated rewriting of the query that contains `SERVICE` calls [4], with query parts that are to be sent to the relevant data sources.

We have implemented the reasoning techniques of Darari et al. [1] using standard Semantic Web frameworks that can process RDF data and SPARQL queries, and reason with RDFS ontologies in a system called CORNER. Moreover, we have built a Web-based demo to show the functionalities of CORNER, which can be accessed at `http://corner.inf.unibz.it/`. While our implementation is based on Apache Jena[1], the approach would also be applicable to other Semantic Web frameworks like OpenRDF Sesame[2]. As a demo for our system, we show various aspects of completeness reasoning in the domain of movies, using the LinkedMDB[3] and DBpedia[4] data sources, which are RDF versions of IMDb and Wikipedia, respectively. Interestingly, IMDb already contains assertions in English about the completeness of cast and crew of movies[5], which are currently still not reflected in its RDF counterpart, LinkedMDB.

2 Motivating Examples

Suppose a moviegoer is interested in finding all movies starring Quentin Tarantino. This information need can be expressed by the SPARQL BGP query:[6]

```
SELECT * WHERE { ?m actor Tarantino }
```

In our demo, CORNER has meta-information about parts of LinkedMDB and DBpedia that are complete. Completeness statements can be represented in two ways: a human-readable abstract syntax, or an RDF syntax, which implements the abstract syntax. Both were developed in [1] and are accepted by CORNER. Abstract completeness statements have the form $\mathit{Compl}(P_1|P_2)$, consisting of two parts: the *pattern* P_1 and the *condition* P_2. The completeness statement specifies that the source contains all data with the pattern shape, provided that in addition they satisfy the condition. To express that a source is complete for "all movies starring Tarantino", we write in the abstract syntax

```
Compl(?m actor Tarantino | true).
```

We attach this statement to LinkedMDB but not to DBpedia, since some information that Tarantino was starred in some movies is actually missing in DBpedia. CORNER then analyzes the query and the statement, and concludes that

[1] `http://jena.apache.org/`
[2] `http://www.openrdf.org/`
[3] `http://linkedmdb.org/`
[4] `http://dbpedia.org/`
[5] As an instance, the page at `http://www.imdb.com/title/tt0105236/fullcredits` about Reservoir Dogs is stated to contain all cast and crew of the movie.
[6] For simplicity, we omit namespaces.

the query over LinkedMDB can be answered completely, while it cannot give such a guarantee for DBpedia.

We imagine that such statements could be part of the meta-information about a data source like the ones provided by VoID descriptions[7]. In fact, completeness statements in RDF syntax can be embedded into VoID descriptions. Alternatively, there could be query hubs that contain such metadata about sources, propose sources suitable for a given query and execute the query over those sources. CORNER demonstrates the second possibility.

Suppose now our moviegoer would also like to see the budget and box-office gross of the movies. This is expressed by the SPARQL BGP query:

```
SELECT *
WHERE { ?m actor Tarantino . ?m budget ?b . ?m gross ?g }
```

Suppose we also have a statement asserting that DBpedia is complete for "the budget and gross of movies starring Tarantino", or in the abstract syntax:

```
Compl(?m budget ?b . ?m gross ?g | ?m actor Tarantino )
```

Note that by the condition, we can express that DBpedia has complete data about budget and box-office gross of movies starring Tarantino, even if in DBpedia Tarantino may not be listed as actor of all such movies. Now, none of the two sources alone is sufficient to answer this new query completely. Suppose as well that we have mappings using the RDFS predicates `subclass` and `subproperty` that associate terms in DBpedia to their LinkedMDB counterparts, if they exist, and vice versa. In this situation, CORNER can rewrite the original query in such a way, using SPARQL `SERVICE` calls, that each source contributes parts of a query for which they are complete. In our example, CORNER sends the subquery asking for movies starring Tarantino to LinkedMDB and the subquery asking for the budget and box-office gross to DBpedia:

```
SELECT *
WHERE {
  SERVICE <http://linkedmdb.org/sparql> { ?m actor Tarantino }
  SERVICE <http://dbpedia.org/sparql> { ?m budget ?b . ?m gross ?g } }
```

3 System Architecture

As shown in Figure 1, CORNER consists of two main components, built on top of the Linked Data layer.

The first component is the user interface (UI), which is developed using the Google Web Toolkit (GWT)[8]. The UI provides users with the possibility to specify what queries they want to check for completeness as well as which completeness statements over which data sources and which RDFS ontologies they want to use for the checking. The second component is the reasoner, the backend of CORNER. The reasoner is implemented using Apache Jena[9]. The

[7] `http://www.w3.org/TR/void/`
[8] `http://www.gwtproject.org/`
[9] `http://jena.apache.org/`

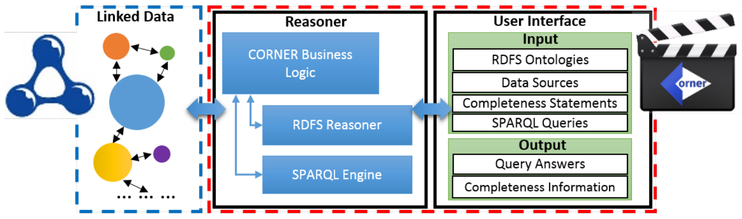


Fig. 1. CORNER Architecture

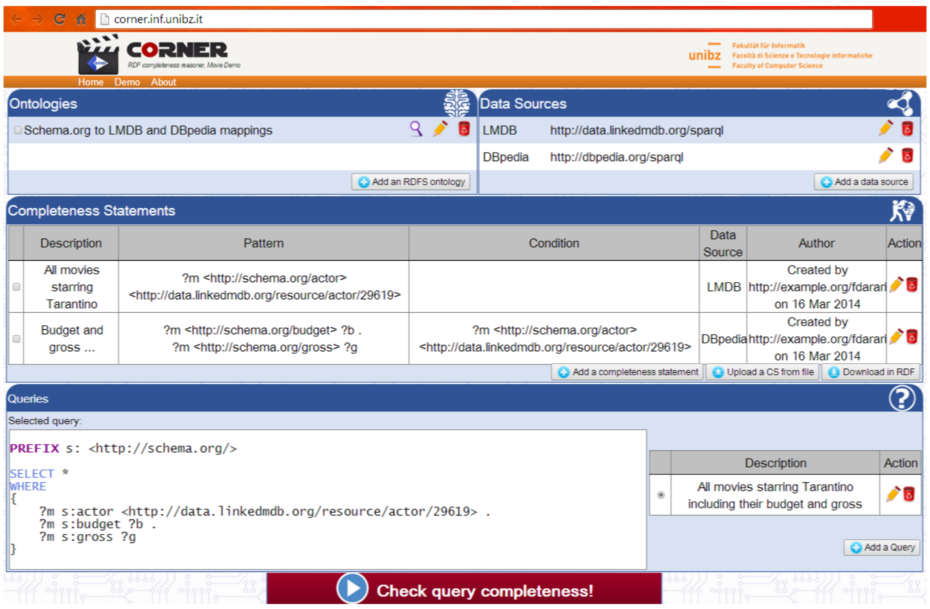
backend performs the completeness reasoning, that is, the query completeness checking based on the inputs. The RDFS reasoner is needed since CORNER takes into account RDFS ontologies. If a query can be ensured to be complete, CORNER rewrites the query into a complete federated version and executes it over Linked Data. For this, the SPARQL engine is necessary. The query results along with the completeness information are given back to the users via the UI.

The processes inside the backend are controlled by the CORNER business logic, which implements the completeness reasoning technique in [1] consisting of the following steps. From the query Q, CORNER generates an initial RDF graph G_Q^i that represents the information needed for answering the query. Moreover, every completeness statement C is translated into a SPARQL `CONSTRUCT` query Q_C. Application of all the queries Q_C to the graph G_Q^i results in a graph G_Q^a, which is a subgraph of G_Q^i and represents the parts of the query for which data are complete. By evaluating Q over G_Q^a, CORNER tests whether the complete data are sufficient to answer Q. Finally, if Q can be answered completely, based on the data sources information of the completeness statements that contribute to generate G_Q^a, CORNER distributes the query parts of Q to their suitable, complete data sources.

4 Demo Description

From the CORNER homepage, users may add RDFS ontologies, data sources, completeness statements of a specific data source, and queries, in addition to those already there. There is a panel in CORNER for each type of information. There are also the options to upload and download CORNER completeness statements in RDF in order to embed them into VoID descriptions of data sources. When adding a new completeness statement, users see a pop-up window where they can specify patterns, conditions, the data source where the statement holds, the author and a description of the completeness statement. When checking the completeness of a query, CORNER displays a pop-up window comprising completeness information about the query, the query results, the debugging information, the ontologies used in the reasoning, a federated rewriting of the query, and the author information for each completeness statement.

Figure 2 shows the example of the query about budget and box-office gross of movies starring Quentin Tarantino, mentioned above. We first specify the



corner.inf.unibz.it

CORNER
RDF completeness reasoner, Movie Demo

unibz Fakultät für Informatik Facoltà di Scienze e Tecnologie informatiche Faculty of Computer Science

Home Demo About

Ontologies

Schema.org to LMDB and DBpedia mappings

Add an RDFS ontology

Data Sources

LMDB	http://data.linkedmdb.org/sparql
DBpedia	http://dbpedia.org/sparql

Add a data source

Completeness Statements

	Description	Pattern	Condition	Data Source	Author	Action
	All movies starring Tarantino	?m <http://schema.org/actor> <http://data.linkedmdb.org/resource/actor/29619>		LMDB	Created by http://example.org/fdarari on 16 Mar 2014	
	Budget and gross ...	?m <http://schema.org/budget> ?b . ?m <http://schema.org/gross> ?g	?m <http://schema.org/actor> <http://data.linkedmdb.org/resource/actor/29619>	DBpedia	Created by http://example.org/fdarari on 16 Mar 2014	

Add a completeness statement | Upload a CS from file | Download in RDF

Queries

Selected query:

```
PREFIX s: <http://schema.org/>

SELECT *
WHERE
{
    ?m s:actor <http://data.linkedmdb.org/resource/actor/29619> .
    ?m s:budget ?b .
    ?m s:gross ?g
}
```

	Description	Action
●	All movies starring Tarantino including their budget and gross	

Add a Query

Check query completeness!

Fig. 2. CORNER Homepage

SPARQL query in the query panel of the Web UI. Then, in the ontology panel, we specify which ontologies we want to use. In this case, we only need to activate the mapping ontology for LinkedMDB and DBpedia. After that, in the completeness statements panel, we select the statements about data sources to be used for query completeness checking. The figure shows the two completeness statements we mentioned above.

To start completeness reasoning, the user has to click the execution button at the bottom of the UI. Now, CORNER returns to the user the query results and information stating that the completeness of the query can be guaranteed. CORNER also provides debugging information about the completeness reasoning and the federated rewriting of the query that was executed over the data sources.

Acknowledgments. This work has been partially supported by the project "MAGIC: Managing Completeness of Data" funded by the province of Bozen-Bolzano, and the European Master's Program in Computational Logic (EMCL).

References

1. Fariz Darari, Werner Nutt, Giuseppe Pirrò, and Simon Razniewski. Completeness statements about RDF data sources and their use for query answering. In *ISWC*, 2013.
2. A.Y. Levy. Obtaining complete answers from incomplete databases. In *PVLDB*, 1996.
3. S. Razniewski and W. Nutt. Completeness of queries over incomplete databases. In *PVLDB*, 2011.
4. S. Harris and A. Seaborne. SPARQL 1.1 query language. Technical report, W3C, 2013.