SPARQLoid - a Querying System using Own Ontology and Ontology Mappings with Reliability

Takahisa Fujino[1] and Naoki Fukuta[2]

[1] Graduate School of Informatics, Shizuoka University
[2] Faculty of Informatics, Shizuoka University
{gs12033@s, fukuta@cs}.inf.shizuoka.ac.jp

Abstract. The heterogeneity of ontologies on the web of data is a crucial problem. This paper presents an application called SPARQLoid that uses a query rewriting method to enable users to query any SPARQL endpoint with the user's own ontology, even when their ontology mappings are not reliable enough. Often ontology matching methods produce mappings with their reliability degree. Based on the given reliability degrees of those mappings, SPARQLoid allows users to query data in the target SPARQL endpoints by using their own (or a specified certain) ontology under a control of sorting order based on their mapping reliability. In this paper, we show a brief demonstration and a comparison to related work.

Keywords: SPARQL, heterogeneous ontology, query translation.

1 Introduction

In this paper, we propose SPARQLoid[3], which can add functionality to SPARQL queries for utilizing the reliability degree in mappings. SPARQLoid enables users to make a query without using the ontologies prepared for the target endpoints; it can also control the resulting outputs based on the reliability degrees in the mappings.

Our developed system uses a query rewriting approach[4] to allow translated SPARQL queries to be reused in user applications. In SPARQLoid, users can use special primitives to specify the criteria to control the results (e.g., controlling sorting orders, placing a threshold to remove unreliable results, etc., see Table 1) based on the mapping reliability to the concepts used in the query. To shorten the generated query strings and to avoid storing large amounts of mapping data in the user applications, the system can generate a query that refers to an external endpoint that stores necessary ontology mappings.

Figure 1 shows our SPARQLoid architecture. We implemented SPARQLoid using Jena 2.6.4 [5] and ARQ 2.8.8 in Jena. The translator engine can use ontology alignment data that are produced by a standard Ontology Alignment API 4.4[6].

[3] A demo video is available at http://whitebear.cs.inf.shizuoka.ac.jp/sparqloid-demo/
[4] A preliminary idea about this approach is presented in [1].
[5] http://jena.apache.org/about_jena/about.html
[6] http://alignapi.gforge.inria.fr/

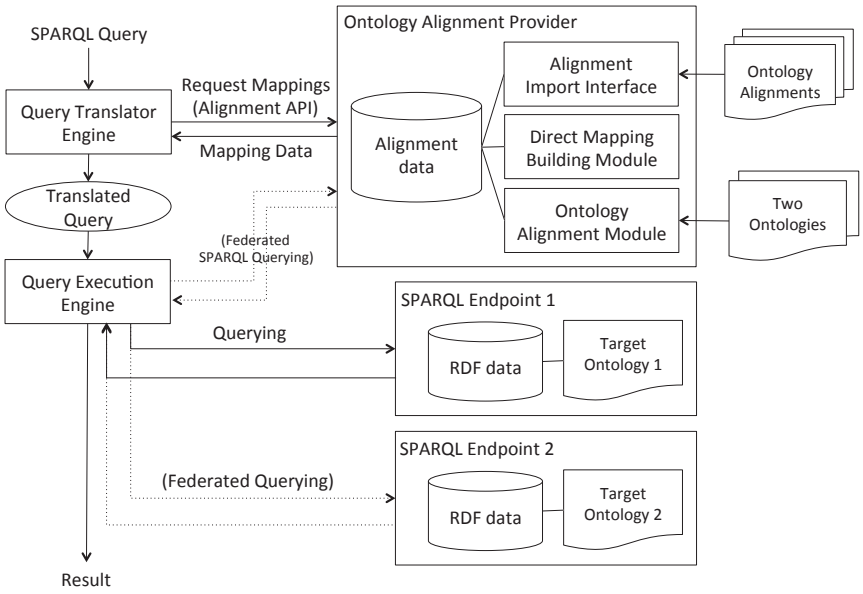

Fig. 1. SPARQLoid Architecture

2 Example

Figure 2 shows a working example of SPARQLoid. First, the user chooses ontology(s) that will be used in the query (Fig. 2(1)). Then, the user enters the endpoint(s) to retrieve data by the query (Fig. 2(2)). The user can see the ontologies that have been already registered in SPARQLoid and confirm whether appropriate mappings exist (Fig. 2(3)) or generate new mappings by APIs(Fig. 2(4)). In the next tab (Figure 2(5)), the user writes a query based on the ontologies chosen in the first step. The query translator engine translates the entered query into a translated query that is written in ordinary SPARQL syntax (Fig. 2(6)). The translated query can be executed on the system (Fig. 2(7)). Furthermore, the user can get programming-friendly code (e.g., Java, PHP, etc.) of the query (Fig. 2(8)).

3 Related Work and Comparison

Mosto[2, 3] is a tool to perform data translation using automatically generated SPARQL executable mappings. The SPARQL-RW[4] framework provides transparent query access over mapped RDF datasets by using Expressive and Declarative Ontology Alignment Language (EDOAL)[7]. The R2R framework [5] au-

[7] http://alignapi.gforge.inria.fr/edoal.html

Expressions in RANKING	Ranking Criterion
A+B+C	sim=a+b+c
max(A)+max(B)+max(C)	sim=max(a)+max(b)+max(c), max(a) means maximum reliability degree when multiple mappings are available for a.
A=0.6,B=0.4,C=0.3	sim = a*0.6+b*0.4+c*0.3
A>B>C	a is used as the primary key for sorting, b is used as the sub-key, and c is used as the sub-subkey
Expressions in THRESHOLD	**Filtering Criterion**
A=0.3,B=0.2,C=0.4	Solutions using a $>$ 0.3, b $>$ 0.2, c $>$ 0.4
max(A)=0.5, max(B)=0.4,max(C)=0.6	Solutions with max(a) $>$ 0.5, max(b) $>$ 0.4, max(c) $>$ 0.6
A:3,B:3,C:3	At most top 3 relevant mappings will be used for A, B, and C

Table 1. Example Syntax of RANKING and THRESHOLD clauses

	SPARQLoid	Mosto[2, 3]	SPARQL-RW[4]	R2R[5]
ability to generate SPARQL-compatible queries	✓		✓	
use of mapping reliability	✓			✓
control syntax for mapping reliability	✓			
ability to translate data		✓		✓
ability to integrate data		✓		✓
ability to generate mappings	*	✓		
support for complex mappings		✓	✓	✓
ability to publish mappings	**			✓
support for interlinking mappings				✓
application embeddability	✓		✓	✓

✓implemented, * partially implemented, ** under development

Table 2. Comparison to Related Work

tomatically discovers and publishes mappings among Linked Data sources for better integration. Table 2 briefly compares our work and the above works.

References

1. Fujino, T. and Fukuta, N.: A SPARQL Query Rewriting Approach on Heterogeneous Ontologies with Mapping Reliability, *The 3rd IIAI International Conference on e-Services and Knowledge Management (IIAI ESKM 2012)*, (2012). (to appear)
2. Rivero, C., Hernandez,I., Ruiz,D., and Corchuelo, R.: Mosto: Generating SPARQL Executable Mappings Between Ontologies, *The 30th International Conference on Conceptual Modeling Demos and Posters*, (2011).
3. Rivero, C., Hernandez,I., Ruiz,D., and Corchuelo, R.: Generating SPARQL Executable Mappings to Integrate Ontologies, *Proc. of the 30th International Conference on Conceptual Modeling*, pp. 118–131, (2011).
4. Makris, K., Bikakis, N., Gioldasis, N., and Christodoulakis, S.: SPARQL-RW: Transparent Query Access over Mapped RDF Data Sources, *The15th International Conference on Extending Database Technology (EDBT2012)*, (2012).
5. Bizer, C. and Schultz, A.: The R2R Framework: Publishing and Discovering Mappings on the Web, *The 1st International Workshop on Consuming Linked Data (COLD 2010)*, (2010).

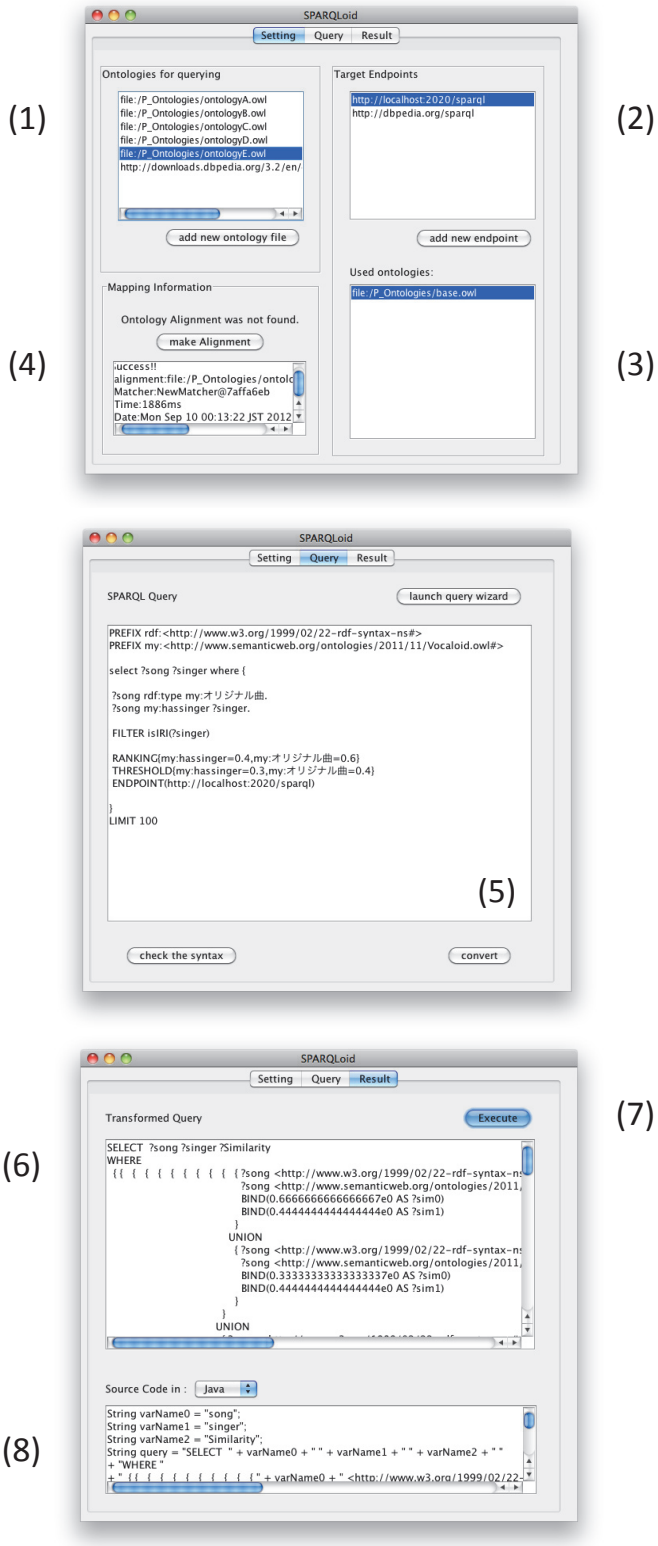

Fig. 2. System Overview