Towards Discovering Ontological Models from Big RDF Data

Carlos R. Rivero, Inma Hernández, David Ruiz, and Rafael Corchuelo

University of Sevilla, Spain
{carlosrivero,inmahernandez,druiz,corchu}@us.es

Abstract. The Web of Data, which comprises web sources that provide their data in RDF, is gaining popularity day after day. Ontological models over RDF data are shared and developed with the consensus of one or more communities. In this context, there usually exist more than one ontological model to understand RDF data, therefore, there might be a gap between the models and the data, which is not negligible in practice. In this paper, we present a technique to automatically discover ontological models from raw RDF data. It relies on a set of SPARQL 1.1 structural queries that are generic and independent from the RDF data. The output of our technique is a model that is derived from these data and includes the types and properties, subtypes, domains and ranges of properties, and minimum cardinalities of these properties. Our technique is suitable to deal with Big RDF Data since our experiments focus on millions of RDF triples, i.e., RDF data from DBpedia 3.2 and BBC. As far as we know, this is the first technique to discover such ontological models in the context of RDF data and the Web of Data.

Keywords: Ontological models, Web of Data, RDF, SPARQL 1.1.

1 Introduction

The goal of the Semantic Web is to endow the current Web with metadata, i.e., to evolve it into a Web of Data [23, 28]. Currently, there is an increasing popularity of the Web of Data, chiefly in the context of Linked Open Data, which is a successful initiative that consists of a number of principles to publish, connect, and query data in the Web [3]. Sources that belong to the Web of Data focus on several domains, such as government, life sciences, geography, media, libraries, or scholarly publications [14]. These sources offer their data using the RDF language, and they can be queried using the SPARQL query language [1].

The goal of the Web of Data is to use the Web as a large database to answer structured queries from users [23]. One of the most important research challenges is to cope with scalability, i.e., processing data at Web scale, usually referred to as Big Data [5]. Additionally, sources in the Web of Data are growing steadily, e.g., in the context of Linked Open Data, there were roughly 12 such sources in 2007 and, as of the time of writing this paper, there exist 326 sources [19]. Therefore, the problem of Big Data increases due to this large amount of sources.

Ontological models are used to model RDF data, and they comprise types, data properties, and object properties, each of which is identified by a URI [1]. These models are shared and developed with the consensus of one or more communities [26], which define a number of inherent constraints over the models, such as subtypes, the domains and/or ranges of a property, or the minimum and maximum cardinalities of a property.

It is important to notice that, in "traditional" information systems, developers first need to create a data model according to the user requirements, which is later populated. Contrarily, in web-of-data information systems, data can exist without an explicit model; even more, several models may exist for the same set of data. Therefore, in the context of the Web of Data, we cannot usually rely on existing ontological models to understand RDF data since there might be a gap between the models and the data, i.e., the data and the model are usually devised in isolation, without taking each other into account [11]. Furthermore, RDF data may not satisfy a particular ontological model related to these data, which is mandatory to perform a number of tasks, such as data integration [20], data exchange [25], data warehousing [12], or ontology evolution [9].

We have identified two common situations in practice in which the gap between ontological models and RDF data is not negligible, namely:

- Languages to represent ontological models provide constructs to express user-defined constraints that are local, i.e., a user or a community can add them to adapt existing models to local requirements [7]. For instance, the ontological model of DBpedia 3.7 [4], which is a community effort to make the data stored at Wikipedia accessible using the Linked Open Data principles, defines a property called $almaMater$ that has type $Person$ as domain, and type $EducationalInstitution$ as range. It is not difficult to find out that this property has also types $City$ and $Country$ as ranges in the RDF data. As a conclusion, there are cases in which RDF data may not be modelled according to existing ontological models, i.e., the data may not satisfy the constraints of the models.
- Some ontological models simply define vocabularies with very few constraints. Therefore, it is expected that users of these ontological models apply them in different ways [27]. For instance, the ontological model of DBpedia 3.7 defines a property called *similar* that has neither domain nor range. In the RDF data, we observe that this property has two different behaviours: one in which type $Holiday$ is the domain and range of the property, and another one in which type $Place$ is the domain and range of the property. As a conclusion, different communities may generate a variety of RDF data that rely on the same ontological models with disparate constraints.

In this paper, we present a technique to automatically discover ontological models from raw RDF data. It aims to solve the gap between the models and the data. Our technique assumes that the model of a set of RDF data is not known a priori, which is a common situation in practice in the context of the Web of Data. To perform this discovery, we rely on a set of SPARQL 1.1 structural

queries that are generic and independent from the RDF data, i.e., they can be applied to discover an ontological model in any set of RDF data.

The output of our technique is a model that includes the types and properties, subtypes, domains and ranges of properties, and minimum cardinalities of these properties. However, currently, we are not able to compute a number of constraints, such as subproperties, maximum cardinalities, or unions of types. Our technique is suitable to deal with Big RDF Data since our experiments focus on millions of RDF triples, i.e., RDF data from DBpedia 3.2 and BBC. To the best of our knowledge, this is the first technique to discover such ontological models in the context of RDF data and the Web of Data.

This paper is organised as follows: Section 2 describes the related work; Section 3 presents our technique to discover ontological models from RDF data that relies on a set of SPARQL 1.1 queries; Section 4 describes two experiments to discover the ontological models behind the RDF data of DBpedia 3.2 and BBC; finally, Section 5 recaps on our main conclusions.

2 Related Work

Research efforts on the automatic discovery of data models have focused on the Deep Web, in which web pages are automatically produced by filling web templates using the data of a back-end database [13]. In the context of the Web of Data, current research efforts assume that RDF data satisfy all of the constraints of the ontological models that model them; however, this situation is not so common in practice.

There are a number of proposals in the literature that aim to discover types from instances, i.e., a particular instance has a particular type. The vast majority of these proposals discover different types in web sites by clustering web pages of the same type [6, 10, 16, 21]. Mecca et al. [21] developed an algorithm for clustering search results of web sites by type that discovers the optimal number of words to classify a web page. Blanco et al. [6] devised a technique to automate the clustering of web pages by type in large web sites. The authors do not rely on the content of web pages, but only on the URLs. Hernández et al. [16] devised a technique similar in spirit to [6] technique, but using a smaller subset of web pages as the training set to automatically cluster the web pages. As a conclusion, these proposals are only able to discover types and no relationships amongst them, such as data properties, object properties, or subtypes. Giovanni et al. [10] aimed to automatically discover the untyped entities that DBpedia comprises, and they proposed two techniques based on induction and abduction.

There exist a number of proposals that are able to automatically discover the data models that are implicit in the semi-structured data that is rendered in a web page. The vast majority of these proposals focus on automating the extraction of information from these web pages [2, 8, 17], and the data models that they are able to discover comprise types and relationships amongst those types. As a conclusion, these proposals are not able to automatically infer the whole data model of the back-end database, but only a part of it.

Other proposals allow to discover complex data models that include types, properties, domains and ranges. These proposals are not fully-automated since they require the intervention of a user. Tao et al. [30] presented a proposal that automatically infers a data model by means of a form, and they deal with any kind of form, not necessarily HTML forms. In this case, the user is responsible for handcrafting these forms; unfortunately, this approach is not appealing since integration costs may be increased if the user has to intervene [22]. Furthermore, this proposal is not able to deal with subtypes.

Hernández et al. [15] devised a proposal that deals with discovering the data model behind a web site. This proposal takes a set of URL patterns that describe the types in a web site as input. Its goal is to discover properties amongst the different types that, in addition to the URL patterns of types, form a data model. The main drawback of this proposal is that it requires the intervention of the user: the final data model comprises a number of anonymous properties and the user is responsible for naming them, which may increase integration costs. In addition, this proposal is not able to discover data properties or subtypes.

Finally, Su et al. [29] developed a fully-automated proposal that discovers an ontological model that is based on the HTML forms of a web site, and the HTML results of issuing queries by means of these forms. In this case, there is no intervention of a user to discover the final ontological model, which is performed by means of a number of matchings amongst the HTML results and the HTML forms. To build the final model, the authors apply nine heuristics, such as "if a matching is unique, a new attribute is created", or "if the matching is n:1, $n+1$ attributes are created". The main drawback of this proposal is that it does not discover subtypes or the name of the properties, i.e., the final model is more a nested-relational model than an ontological model. Note that a nested-relational model is defined by means of a tree that comprises a number of nodes, which may be nested and have a number of attributes, and it is also possible to specify referential constraints that relate these attributes [24].

3 Discovering Ontological Models

We have devised a technique that relies on a number of SPARQL 1.1 queries to discover ontological models from raw RDF data. In this section, we use a running example based on DBpedia, which has undergone several revisions. We focus on a part of DBpedia 3.2 that comprises $2,107,451$ triples, which is a dataset of Big RDF Data.

RDF data comprise triples of two kinds: type and property triples. A triple comprises three elements: the subject, the predicate, and the object, respectively. Both subjects and predicates are URIs, and objects may be URIs or literals. In the rest of this paper, we use a number of prefixes that are presented in Table 1. A type triple relates a URI with a particular type by means of a type predicate, e.g., $(dbpd{:}Clint\_Eastwood, rdf{:}type, dbpo{:}Actor)$ states that Clint Eastwood is an actor. A data property triple relates a URI with a literal using a property, e.g., $(dbpd{:}Clint\_Eastwood, dbpo{:}birthDate, \text{“}1930{-}05{-}31\text{”}\hat{}\hat{}xsd{:}date)$ is

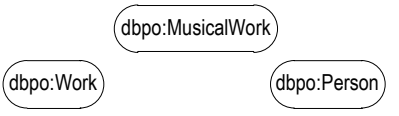
Table 1. Prefixes used throughout the paper

Prefix	URI
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
xsd	`http://www.w3.org/2001/XMLSchema#`
dbpo	`http://dbpedia.org/ontology/`
dbpd	`http://dbpedia.org/resource/`
po	`http://purl.org/ontology/po/`
dc	`http://purl.org/dc/elements/1.1/`

a triple that states that the birth date of Clint Eastwood is May 31, 1930, which is of *xsd*:*date* type. An object property triple relates two URIs by means of a property, e.g., (*dbpd*:$Dirty\_Harry$, *dbpo*:*starring*, *dbpd*:$Clint\_Eastwood$) is a triple stating that film Dirty Harry is starred by Clint Eastwood.

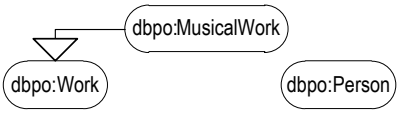
Figure 1 presents a summary of our technique based on the ontological model of DBpedia 3.2: we first discover types and subtypes; then, we discover properties, the domains and ranges of these properties, and their minimum cardinalities. To discover this model, we issue a number of SPARQL 1.1 queries over the RDF data that are also presented in this figure, in which we enclose parameters between $ symbols. In the rest of this section, we describe each of these steps in detail:

1. In the first step, we discover types from the input RDF data, such as *dbpo*:$Person$, *dbpo*:$MusicalWork$, or *dbpo*:$Work$ (see Figure 1a). To discover them, we project the types of all instances without repetition.
2. In the second step, we discover subtypes amongst the previously discovered types. To perform this, we iterate two times over the whole set of types, so, for each pair of types t_1 and t_2, assuming that $t_1 \neq t_2$, we have that t_1 is subtype of t_2 if each instance of type t_1 is also an instance of type t_2. An example is that *dbpo*:$MusicalWork$ is subtype of *dbpo*:$Work$ (see Figure 1b). Note that we use the negation of the query in Figure 1b, i.e., t_1 is subtype of t_2 if the query returns false.
3. In the third step, we discover properties from the input RDF data, such as *dbpo*:*birthDate*, *dbpo*:*starring*, or *dbpo*:*director* (see Figure 1c). We project the predicates that relate all triples without repetition.
4. The fourth step deals with discovering domains, such as the domain of *dbpo*:*starring* is *dbpo*:$Work$ (see Figure 1d). To discover the domains of a property *prop*, we retrieve all triples that have this property as predicate, and we project the types of the subjects in these triples without repetition.
5. The fifth step is similar to the previous step, but we discover ranges instead of domains (see Figure 1e).
6. The sixth step discovers minimum cardinalities of the previously discovered domains and ranges. An example is that the minimum cardinality of *dbpo*:*starring* for domain *dbpo*:$Work$ is zero since there exists, at least,



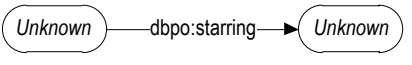

```
SELECT DISTINCT ?t
WHERE {
    ?x  rdf:type    ?t . }
```

(a) First step: discovering types



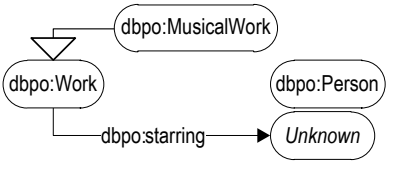

```
ASK {
    ?s  rdf:type $t_1$ .
    FILTER NOT EXISTS {
        ?s rdf:type $t_2$ . } }
```

(b) Second step: discovering subtypes



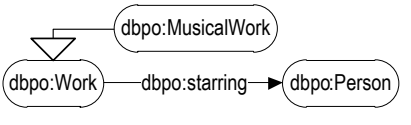

```
SELECT DISTINCT ?p
WHERE {
    ?s  ?p  ?o . }
```

(c) Third step: discovering properties



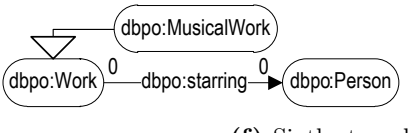

```
SELECT DISTINCT ?d
WHERE {
    ?s   rdf:type  ?d ;
         $prop$    ?o . }
```

(d) Fourth step: discovering domains




```
SELECT DISTINCT ?r
WHERE {
    ?o   rdf:type  ?r .
    ?s  $prop$    ?o . }
```

(e) Fifth step: discovering ranges




```
ASK {
    ?s  rdf:type $type$ .
    FILTER NOT EXISTS {
        ?s  $prop$ ?o . } }
ASK {
    ?o  rdf:type $type$ .
    FILTER NOT EXISTS {
        ?s  $prop$ ?o . } }
```

(f) Sixth step: discovering cardinalities

Fig. 1. Steps of our technique to discover ontological models from RDF data

one instance of *dbpo*:$Work$ that is not related by property *dbpo*:*starring* (see Figure 1f). Another example is that the minimum cardinality of property *dbpo*:*starring* for range *dbpo*:$Person$ is zero since there exists, at least, one instance of $dbpo : Person$ that is not the subject of an instance of property $dbpo : starring$. To perform this, we ask if there is any domain or range instance of a given type *type* related by a particular property *prop*. If this is true, the minimum cardinality is zero. Otherwise, we count the minimum number of instances of type *type* related to property *prop*.

Our technique is not able to discover a number of constraints, but some of them may be addressed, e.g., maximum cardinalities and subproperties. Regarding maximum cardinalities, our technique is able to compute a bound of the cardinality, but not the exact cardinality. For instance, we have computed that the maximum cardinality of property *dbpo*:*starring* for domain *dbpo*:$Work$ is 74, however, this number is not the exact cardinality since it probably allows unbounded instances. Regarding subproperties, we may use a technique similar to the second step to discover subtypes.

4 Experiment Results

We implemented our technique using Java 1.6 and OWLIM Lite 4.2, which comprises an RDF store and a SPARQL query engine. In this experiment, we computed the times taken by our technique to discover the ontological models behind the RDF data of a part of DBpedia 3.2 and BBC. The BBC [18] decided to adhere to the Linked Open Data principles in 2009. They provide ontological models that adhere to these principles to publicise the music and programmes they broadcast in both radio and television.

To compute the times taken by our technique, we ran the experiment on a virtual computer that was equipped with a four-threaded Intel Xeon 3.00 GHz CPU and 16 GB RAM, running on Windows Server 2008 (64-bits), JRE 1.6.0. Furthermore, we repeated the experiment 25 times and computed the maximum values. Table 2 shows our results when applying our technique to DBpedia 3.2 and BBC. The first column of the table stands for the different steps of our technique; the second column deals with the total number of constraints that we have discovered; finally, the third column shows the time in minutes taken by our technique to compute each step.

The total time that our technique took was 31.15 minutes for DBpedia 3.2, which comprises a total number of $2,107,451$ triples, and 2.48 minutes for BBC, which comprises a total number of $7,274,597$ triples. At a first glance, it might be surprising that the time taken for BBC is less than the time taken for DBpedia, since BBC comprises more triples than DBpedia. This is due to the fact that the time of our technique depends on the structural complexity of the discovered ontological model, and it does not depend on the data. Therefore, we may conclude that the structural complexity of the ontological model of DBpedia 3.2 is greater than the structural complexity of the BBC model.

Table 2. Summary of results of discovering ontological models behind RDF data

(a) DBpedia 3.2

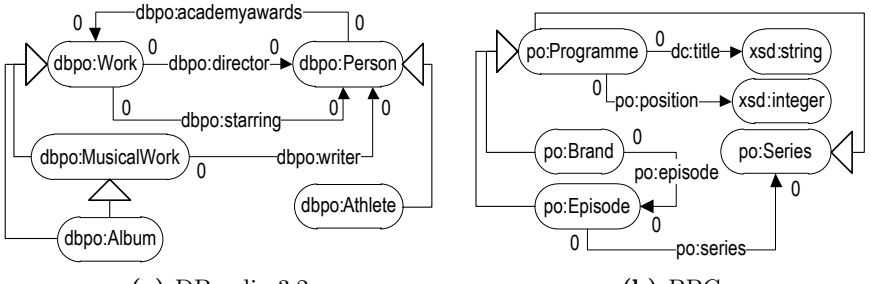
Step	Constr.	Time (min)
Types	91	0.03
Subtypes	328	0.12
Properties	398	0.02
Domains	1,148	16.15
Ranges	1,148	12.35
Cardinalities	4,592	2.48
Total	7,705	31.15

(b) BBC

Step	Constr.	Time (min)
Types	15	0.03
Subtypes	6	0.17
Properties	28	0.05
Domains	39	0.99
Ranges	39	1.09
Cardinalities	156	0.15
Total	283	2.48

Figure 2a shows a part of the ontological model that results from applying our technique to the RDF data of DBpedia 3.2. In this case, the model comprises five types, namely: *dbpo*:*Person*, *dbpo*:*Work*, *dbpo*:*Athlete*, *dbpo*:*MusicalWork*, and *dbpo*:*Album*. In addition to these types, the model comprises four subtype relationships, and four properties with their domains and ranges, namely: *dbpo*:*starring*, *dbpo*:*director*, *dbpo*:*writer*, and *dbpo*:*academyawards*. Finally, the minimum cardinalities for all properties are zero.

Figure 2b shows a part of the model that results from the RDF data of BBC, which comprises four types, namely: *po*:*Programme*, *po*:*Brand*, *po*:*Episode*, and *po*:*Series*. It also comprises three subtype relationships, and four properties with their domains and ranges, namely: *dc*:*title*, *po*:*position*, *po*:*episode*, and *po*:*series*. Note that the minimum cardinalities for all properties are also zero.



(a) DBpedia 3.2 **(b)** BBC

Fig. 2. A part of the ontological models that result from our experiments

5 Conclusions

In the context of the Web of Data, there exists a gap between existing ontological models and RDF data due to the following reasons: 1) RDF data may not satisfy the constraints of the existing ontological models; 2) different communities may

generate a variety of RDF data that rely on the same ontological models with disparate constraints. This gap is not negligible and may hinder the practical application of RDF data and ontological models in other tasks, such as data integration, data exchange, data warehousing, or ontology evolution. To solve this gap, we present a technique to discover ontological models from raw RDF data that relies on a set of SPARQL 1.1 structural queries. The output of our technique is a model that includes types and properties, subtypes, domains and ranges of properties, and minimum cardinalities of these properties.

Acknowledgements. Supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, and TIN2010-09988-E).

References

[1] Antoniou, G., van Harmelen, F.: A Semantic Web Primer. The MIT Press (2008)
[2] Arasu, A., Garcia-Molina, H.: Extracting structured data from web pages. In: SIGMOD Conference, pp. 337–348 (2003)
[3] Bizer, C., Heath, T., Berners-Lee, T.: Linked Data: The story so far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
[4] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - A crystallization point for the Web of Data. J. Web Sem. 77(3), 154–165 (2009)
[5] Bizer, C., Boncz, P., Brodie, M.L., Erling, O.: The meaningful use of Big Data: Four perspectives - four challenges. SIGMOD Record 40(4), 56–60 (2011)
[6] Blanco, L., Dalvi, N.N., Machanavajjhala, A.: Highly efficient algorithms for structural clustering of large websites. In: WWW, pp. 437–446 (2011)
[7] Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H.: Contextualizing ontologies. J. Web Sem. 1(4), 325–343 (2004)
[8] Crescenzi, V., Mecca, G.: Automatic information extraction from large websites. J. ACM 51(5), 731–779 (2004)
[9] Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: Classification and survey. Knowledge Eng. Review 23(2), 117–152 (2008)
[10] Giovanni, A., Gangemi, A., Presutti, V., Ciancarini, P.: Type inference through the analysis of wikipedia links. In: LDOW (2012)
[11] Glimm, B., Hogan, A., Krötzsch, M., Polleres, A.: OWL: Yet to arrive on the Web of Data? In: LDOW (2012)
[12] Glorio, O., Mazón, J.-N., Garrigós, I., Trujillo, J.: A personalization process for spatial data warehouse development. Decision Support Systems 52(4), 884–898 (2012)
[13] He, B., Patel, M., Zhang, Z., Chang, K.C.-C.: Accessing the Deep Web. Commun. ACM 50(5), 94–101 (2007)
[14] Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool (2011)

[15] Hernández, I., Rivero, C.R., Ruiz, D., Corchuelo, R.: Towards Discovering Conceptual Models behind Web Sites. In: Atzeni, P., Cheung, D., Sudha, R. (eds.) ER 2012. LNCS, vol. 7532, pp. 166–175. Springer, Heidelberg (2012)
[16] Hernández, I., Rivero, C.R., Ruiz, D., Corchuelo, R.: A statistical approach to URL-based web page clustering. In: WWW, pp. 525–526 (2012)
[17] Kayed, M., Chang, C.-H.: FiVaTech: Page-level web data extraction from template pages. IEEE Trans. Knowl. Data Eng. 22(2), 249–263 (2010)
[18] Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C., Lee, R.: Media Meets Semantic Web – How the BBC Uses DBpedia and Linked Data to Make Connections. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 723–737. Springer, Heidelberg (2009)
[19] LOD Cloud. Linked Open Data cloud (April 2012), `http://thedatahub.org/group/lodcloud`
[20] Makris, K., Gioldasis, N., Bikakis, N., Christodoulakis, S.: SPARQL-RW: Transparent query access over mapped RDF data sources. In: EDBT (2012)
[21] Mecca, G., Raunich, S., Pappalardo, A.: A new algorithm for clustering search results. Data Knowl. Eng. 62(3), 504–522 (2007)
[22] Petropoulos, M., Deutsch, A., Papakonstantinou, Y., Katsis, Y.: Exporting and interactively querying web service-accessed sources: The CLIDE system. ACM Trans. Database Syst. 32(4), 22 (2007)
[23] Polleres, A., Huynh, D.: Special issue: The Web of Data. J. Web Sem. 7(3), 135 (2009)
[24] Popa, L., Velegrakis, Y., Miller, R.J., Hernández, M.A., Fagin, R.: Translating web data. In: VLDB, pp. 598–609 (2002)
[25] Rivero, C.R., Hernández, I., Ruiz, D., Corchuelo, R.: On benchmarking data translation systems for semantic-web ontologies. In: CIKM, pp. 1613–1618 (2011)
[26] Rivero, C.R., Hernández, I., Ruiz, D., Corchuelo, R.: Generating SPARQL Executable Mappings to Integrate Ontologies. In: Jeusfeld, M., Delcambre, L., Ling, T.-W. (eds.) ER 2011. LNCS, vol. 6998, pp. 118–131. Springer, Heidelberg (2011b)
[27] Rivero, C.R., Schultz, A., Bizer, C., Ruiz, D.: Benchmarking the performance of Linked Data translation systems. In: LDOW (2012)
[28] Shadbolt, N., Berners-Lee, T., Hall, W.: The Semantic Web revisited. IEEE Intelligent Systems 21(3), 96–101 (2006)
[29] Su, W., Wang, J., Lochovsky, F.H.: ODE: Ontology-assisted data extraction. ACM Trans. Database Syst. 34(2), 12 (2009)
[30] Tao, C., Embley, D.W., Liddle, S.W.: FOCIH: Form-Based Ontology Creation and Information Harvesting. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 346–359. Springer, Heidelberg (2009)