IEICE TRANS. INF. & SYST., VOL.E96–D, NO.1 JANUARY 2013

PAPER

Integrating Ontologies Using Ontology Learning Approach

Lihua ZHAO[†a)], ***Nonmember and*** **Ryutaro ICHISE**[†b)], ***Member***

SUMMARY The Linking Open Data (LOD) cloud is a collection of linked Resource Description Framework (RDF) data with over 31 billion RDF triples. Accessing linked data is a challenging task because each data set in the LOD cloud has a specific ontology schema, and familiarity with the ontology schema used is required in order to query various linked data sets. However, manually checking each data set is time-consuming, especially when many data sets from various domains are used. This difficulty can be overcome without user interaction by using an automatic method that integrates different ontology schema. In this paper, we propose a Mid-Ontology learning approach that can automatically construct a simple ontology, linking related ontology predicates (class or property) in different data sets. Our Mid-Ontology learning approach consists of three main phases: data collection, predicate grouping, and Mid-Ontology construction. Experiments show that our Mid-Ontology learning approach successfully integrates diverse ontology schema with a high quality, and effectively retrieves related information with the constructed Mid-Ontology.

key words: *mid-ontology, linked data, Semantic Web, ontology learning, ontology integration*

1. Introduction

The Linking Open Data (LOD) cloud is a collection of Resource Description Framework (RDF) data in <subject, predicate, object> triples [1]. The LOD cloud (as of September 2011) contains 295 data sets mainly categorized into seven domains: cross-domain, geographic, media, life sciences, government, user-generated content, and publications. Things are represented using the Uniform Resource Identifier (URI), and identical or related things are linked with the built-in OWL property *owl:sameAs* [2]. The Web Ontology Language (OWL) is a semantic markup language designed for sharing or publishing ontologies on the Web [3].

Although many applications, such as linked data browsers [4], semantic search engines [5], and some domain specific applications [6], have been developed that demand access to linked data sets, integrating ontology schema or data sets from diverse domains remains a challenging problem [7]. This problem persists because multiple data sets provide different values for the same predicate of an object or provide different terms to represent the same predicate [8]. SPARQL Protocol and RDF Query Language (SPARQL) is a powerful RDF query language that enables Semantic Web users to access linked data [9]. However, users must understand the ontology schema of the data sets to construct SPARQL queries [10]. Learning all the ontology schema is not feasible and is time-consuming, because each data set has a specially designed ontology and thus thousands of distinct ontology predicates might exist. For example, the postal code is represented using *db-onto:postalCode*[*] in DBpedia, and is represented using *geo-onto:postalCode*[**] in Geonames. If these two predicates are integrated together into one ontology predicate, for instance, in the *mo-onto:postalCode*, we can easily search places with the same postal code in both DBpedia and Geonames by only querying with the *mo-onto:postalCode*. Querying with one simple ontology that integrates various ontologies can simplify SPARQL queries and help Semantic Web application developers to easily understand ontology schema so that they may retrieve rich information from various linked data sets.

To solve this problem, an automatic method to construct a simple ontology that integrates ontology schema from diverse domain data sets must be developed. Ontology alignment, or ontology matching is commonly used to find correspondences between ontologies to solve the ontology heterogeneity problem [11]. Ontology learning technology can automate the ontology construction process from structured, semi-structured or unstructured data [12]. An ontology learning cycle that includes ontology design, ontology learning, and validation phases is introduced in [13]. However, the majority of the research on ontology learning technology focuses on text files [14], [15]. To adapt to the LOD data sets, we designed an automatic ontology learning approach, which can construct an integrated ontology from various linked data sets.

In this paper, we present the automatic Mid-Ontology learning approach, which includes ontology manipulations such as ontology term extraction, ontology matching, and ontology integration. Ontology integration is defined as a process that generates a single ontology from different existing ontologies [16]. An automatically constructed ontology is called a Mid-Ontology, and integrates related ontology predicates from diverse linked data sets.

This paper is organized as follows. We start with the introduction of our automatic Mid-Ontology learning approach, which involves data collection, predicate group-

Manuscript received April 6, 2012.
Manuscript revised August 7, 2012.

[†]The authors are with National Institute of Informatics, Tokyo, 101–8430 Japan.
a) E-mail: lihua@nii.ac.jp
b) E-mail: ichise@nii.ac.jp

DOI: 10.1587/transinf.E96.D.40

[*]db-onto: http://dbpedia.org/ontology/
[**]geo-onto: http://www.geonames.org/ontology#

Table 1 Collected data based on the db:Hamburg.

Predicate	Object
db − prop : *name*	"Free and Hanseatic City of Hamburg"@en
db − prop : *population*	1769117
db − onto : *populationTotal*	1769117
.....	.....
geo − onto : *officialName*	"Hamburg"@de
geo − onto : *officialName*	"Free and Hanseatic City of Hamburg"@en
geo − onto : *population*	1739117
.....	.....
skos : *prefLabel*	"Hamburg (Germany)"@en
nyt − prop : *first_use*	2006-12-12
nyt − prop : *latest_use*	2010-05-14

ing, and Mid-Ontology construction. Next, we evaluate our Mid-Ontology learning approach from five different perspectives: the effectiveness of data reduction, the quality of Mid-Ontology, comparison results with previous work, the effectiveness of information retrieval with a SPARQL query, and the characteristics of the integrated predicates. Then, we discuss the advantages of our Mid-Ontology learning approach and its possible applications, followed by some previous related research and a comparison with our approach. In the last section, we present our conclusion and future work.

2. Mid-Ontology Learning Approach

In the LOD cloud, data sets are linked with *owl:sameAs* at an instance level, but few links exist at the class or property level. Although the RDF link types owl:equivalentClass and owl:equivalentProperty are designed to indicate that two classes or properties refer to the same concept, there are only few links at a class level or property level [9]. Hence, whenever linked data sets are queried with SPARQL, the predicates of the ontology schema must be manually learned, an infeasible task if there are thousands of distinct predicates.

Our aim is to automatically construct a simple ontology that integrates ontology schema from various linked data sets. By collecting linked instances, we can identify different predicates that indicate identical or related information. For example, Table 1 shows the collected <predicate, object> pairs of instances that indicate the place "Hamburg" from DBpedia [17], Geonames, and NYTimes, where these instances are linked with *owl:sameAs*. In Table 1, there are three distinct predicates that indicate "Hamburg," each belonging to a different data set. If we can integrate these related predicates into one predicate, we can query the entire data sets with one single predicate that indicates the name of a place.

In this section, we describe the architecture of our Mid-Ontology learning approach, as shown in Fig. 1. The architecture of our approach comprises three phases: data collection, predicate grouping, and Mid-Ontology construction.

2.1 Data Collection

Although the SameAs (*owl:sameAs*) link is designed to link identical things, it also links related or similar things in published linked data sets [18]. Hence, we can find identical or related information if we investigate instances linked with *owl:sameAs*. In this section, we describe our data collection phase in three steps: extract data linked with *owl:sameAs*, remove noisy instances of the core data set, and collect predicates and objects to construct the final data set for our Mid-Ontology learning.

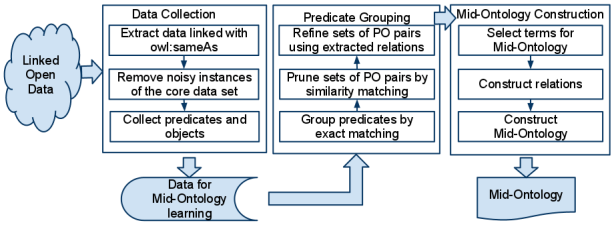

Fig. 1 Architecture of our Mid-Ontology learning approach.

2.1.1 Extract Data Linked with *owl:sameAs*

To extract data linked with *owl:sameAs*, we have to select a core data set, which serves as a hub of several linked data sets. A good core data set should have inward or outward links to other data sets from diverse domains. After the core data set selection, we also collect all the instances that have the SameAs links with instances in the core data set.

For instance, if we select DBpedia as the core data set, then we select all the DBpedia instances that have SameAs links with other data sets, such as Geonames and NYTimes. Table 1 shows an example of the collected data based on the DBpedia instance db:Hamburg[†]. Because both the instances *http://data.nytimes.com/N78784173230527526471* and *http://sws.geonames.org/2911297/* are connected with db:Hamburg, we also collect the contents of these two URIs.

2.1.2 Remove Noisy Instances of the Core Data Set

We say an instance is *noisy*, if none of the triples of the instance contains information that can represent the characteristics of the instance. For instance, if all the triples of an instance are SameAs links or broken links, we cannot learn any information that can represent the characteristics of an instance. We remove these noisy instances of the core data set before collecting predicates and objects.

2.1.3 Collect Predicates and Objects

Data collection is based on each retrieved instance of the core data set, from which we collect all the <predicate, object> pairs of the instance and of the instances connected with the instance. Hereafter, we use *PO* to represent a <predicate, object> pair.

In this *PO* collection step, we do not collect the triples with the SameAs link, because we have already collected

[†]db: http://dbpedia.org/resource/

triples of the linked instances. Furthermore, for the instances of the core data set, if there is any link to another instance of the core data set, we also collect triples from the linked instance. In addition, if an instance has a redirection to another instance, we also collect triples from the redirected instance, which normally contains richer information than the original instance. The collected data consists of *PO* pairs based on each instance of the core data set, which is used for the predicate grouping phase.

2.2 Predicate Grouping

The grouping of related predicates from different ontology schema is critical to Mid-Ontology construction, because there exist many similar ontology predicates that represent the same thing. The predicate grouping phase consists of three main steps: group predicates by exact matching, prune sets of *PO* pairs by similarity matching, and refine sets of *PO* pairs using extracted relations such as domain and range of the predicates.

2.2.1 Group Predicates by Exact Matching

The first step in predicate grouping is to create the initial sets of *PO* pairs that share identical information or the same object by the exact string matching method. The collected data set is based on each instance of the selected core data, which consists of *PO* pairs. We perform a pairwise comparison of PO_i and PO_j and create the initial sets $S_1, S_2, \ldots, S_k$ by checking whether they share an identical predicate or object. Here, S is a set of *PO* pairs.

For example, in Table 1, both geo-onto:officialName and db-prop:name[†] have the same object, "Free and Hanseatic City of Hamburg"@en, and the predicate geo-onto:officialName has another object, "Hamburg"@de. Hence, these two predicates and objects are grouped together to create an initial set. After creating initial sets for all the *PO* pairs, we create initial sets for each *PO* pair that has not yet been grouped. For instance, nyt-prop:first_use[††] is in an initial set by itself because no predicate has the same object "2006-12-12" and no identical predicate exists in the data.

2.2.2 Prune Sets of *PO* pairs by Similarity Matching

The second step in predicate grouping is pruning the initial sets by string-based similarity matching and knowledge-based similarity matching, which are commonly used to match ontologies at the concept level [19]. We observed that some of the same values that are written in different languages and some semantically identical words, such as U.K. and United Kingdom, may be ignored in the exact matching step. Therefore, similarity matching is required to group semantically similar predicates.

In our approach, we adopted four string-based similarity measures, namely, prefix, suffix, Levenshtein distance, and n-gram, as introduced in [20], [21], and nine knowledge-based similarity measures [22], namely, LCH, RES, HSO, JCN, LESK, PATH,WUP, LIN, and VECTOR, which are based on WordNet (a large lexical database of English [23]). String-based similarity measures are applied to compare objects of predicates, because objects may contain URIs instead of lexical labels or phrases. However, knowledge-based similarity measures are applied to compare pre-processed terms of predicates because the terms of predicates are more likely to have semantic meanings that can be recognized as a concept. In the following, the term T_S indicates the pre-processed terms of the predicates in S, the term O_S indicates the objects stored in S, and the term P_S indicates the predicates stored in S.

To extract the terms of predicates, we pre-process each predicate of the *PO* pairs by performing natural language processing (NLP), which includes tokenizing terms, removing stop words, and stemming terms using the porter stemming algorithm [24]. NLP is a key method for the data preprocessing phase, in which terms are extracted from ontologies; this method helps improve the performance of ontology building [25].

$Sim(S_i, S_j)$ is the similarity between S_i and S_j, which is calculated using the formula:

$$Sim(S_i, S_j) = \frac{StrSim(O_{S_i}, O_{S_j}) + WNSim(T_{S_i}, T_{S_j})}{2}$$

where $StrSim(O_{S_i}, O_{S_j})$ is the average of the four string-based similarity values and $WNSim(T_{S_i}, T_{S_j})$ is the average of the nine applied WordNet-based similarity values. For WordNet-based similarity measures, we do not count on the term pairs that have no similarity value returned from WordNet-based similarity measures. Because if we count on them, it means that we treat the similarity value as zero. However, no returned similarity value from WordNet-based similarity measures does not mean the similarity is zero.

If $Sim(S_i, S_j)$ is higher than a predefined similarity threshold, we consider that these two initial sets share similar predicates, and we merge these two sets. After comparing all the pairwise initial sets, we remove the initial set S_i if it has not been merged during this pruning process and has only one *PO* pair.

Here, we show how to calculate the similarity between two initial sets S_i and S_j, where these two initial sets are created based on Table 1. Suppose S_i includes db-prop:population and db-onto:populationTotal with the object "1769117", and set S_j includes the predicate geo-onto:population with the object "1739117". T_{S_i} includes "population" and "total", while T_{S_j} includes "population". Here, O_{S_i} is "1769117" and O_{S_j} is "1739117".

We performed four string-based similarity measures, including prefix, suffix, Levenshtein, and n-gram, on the two objects, "1769117" and "1739117". The similarity values for prefix, suffix, Levenshtein, and n-gram are 0.29, 0.56, 0.4, and 0.86, respectively. Hence, the string-based similarity $StrSim(O_{S_i}, O_{S_j})$ is 0.5275, which is the average of the 4

[†]db-prop: http://dbpedia.org/property/

[††]nyt-prop: http://data.nytimes.com/elements/

Table 2 WordNet-based similarity on pairwise terms.

Pairwise Terms	LCH	RES	HSO	JCN	LESK	PATH	WUP	LIN	VECTOR
population, population	1	1	1	1	1	1	1	1	1
population, total	0.4	0	0	0.06	0.03	0.11	0.33	0	0.06

string-based similarity values. Table 2 shows the WordNet-based similarity values of the pairwise terms in S_i and S_j. $WNSim(T_{S_i}, T_{S_j})$ is 0.5825, which is the average of the 15 similarity values returned by WordNet-similarity measures, as listed in Table 2. Hence, the final similarity $Sim(S_i, S_j)$ is 0.555, which is the average of 0.5275 and 0.5825. If this value is higher than the predefined similarity threshold, we merge S_i and S_j. In this work, we set the default similarity threshold to 0.5.

The pruned sets of *PO* pairs are created by performing pairwise similarity measures on the initial sets and are passed to the refining process.

2.2.3 Refine Sets of *PO* Pairs Using Extracted Relations

The final step of predicate grouping is to split the predicates of each pruned S_i according to the relations of rdfs:domain and rdfs:range [26]. Even though the objects or terms of predicates are similar, the predicates may belong to different domains or ranges. For further refinement, we determine the frequency of each pruned S_i in all of the data and keep any S_i that appears with a frequency that is higher than the predefined frequency threshold. This refining process is applicable to any type of objects because we only consider the domain and range information. The final refined sets of *PO* pairs are passed to the next phase, Mid-Ontology construction.

2.3 Mid-Ontology Construction

According to the refined sets of *PO* pairs, we construct the Mid-Ontology with automatically selected terms and a specially designed predicate.

2.3.1 Select Terms for Mid-Ontology

To perform automatic term selection, we pre-process all the terms of the predicates in each set by tokenization, stop word removal, and stemming. During the pre-process, non-literal terms and stop words are removed because they can not convey meanings. We also keep the original terms because sometimes a single word is ambiguous when it is used to represent a set of terms. For example, "area" and "areaCode" have different meanings but may have the same frequency because the former is extracted from the latter. Hence, when two terms have the same frequency, we choose the longer one. The predicate mo-onto:Term is designed to represent a class term, where the "Term" is automatically selected.

2.3.2 Construct Relations

We designed a predicate mo-prop:hasMembers to link sets of predicates with the Mid-Ontology classes. This predicate indicates that a set of integrated predicates are members of a Mid-Ontology class. We use the relation mo-prop:hasMembers instead of the existing owl:equivalentClass or owl:equivalentProperty in order to reduce the number of triples to connect the integrated predicates. For example, if the number of integrated predicates in a set S_i is n, we need $n * (n - 1)$ triples to connect all the pairs of predicates using owl:equivalentClass or owl:equivalentProperty. However, by connecting with our Mid-Ontology classes, we only need n triples with mo-prop:hasMembers.

2.3.3 Construct Mid-Ontology

A Mid-Ontology is automatically constructed with refined sets of integrated predicates, automatically selected terms, and a designed predicate mo-prop:hasMembers, which links sets of predicates and Mid-Ontology classes.

2.4 Implementation

Many Semantic Web tools are developed to help researchers query linked data, publish linked data, or manage enormous data sets. Virtuoso[†] is a high-performance server that supports the storage of a large RDF data, provides a SPARQL endpoint, and supports the creation of RDF models [27]. Therefore, we used Virtuoso to store linked data sets and queried SPARQL examples for experiments. A Virtuoso Jena RDF Data Provider is also provided, enabling Java applications to directly query the Virtuoso RDF data through Jena RDF Frameworks.

For knowledge-based similarity matching, we used WordNet::Similarity[††] [22], which is implemented in Perl. Several WordNet-based similarity measuring algorithms are implemented in this tool. If two terms are identical, we return 1, which is the maximum similarity value; otherwise, we apply WordNet-based similarity measures. The similarity measures JCN, PATH, WUP, LIN, and VECTOR return normalized values between zero and one. However, the similarity values of LCH, RES, HSO, and LESK are not normalized. To normalize the similarity values of LCH, RES, HSO, and LESK measures, we divide the returned values by the maximum value that can be obtained from all the pairwise terms in the collected data set. The normalized similarity

[†]http://virtuoso.openlinksw.com/
[††]http://wn-similarity.sourceforge.net/

Sim_{alg} is calculated using the formula:

$$Sim_{alg} = \frac{WordNet_{alg}}{Max_{alg}}$$

where $WordNet_{alg}$ indicates the returned value from WordNet::Similarity tool and Max_{alg} indicates the maximum value we obtained from the $WordNet_{alg}$. The Max_{alg} of LCH, RES, HSO, and LESK are 3.7, 10, 16, and 5.6, respectively.

3. Experimental Evaluation

In this section, we first introduce the experimental data used in our experiments. Then we evaluate the Mid-Ontology learning approach from five different perspectives: the effectiveness of data reduction, the quality of constructed Mid-Ontology with different combinations of data sets, the comparison of the Mid-Ontologies created in our current work with previous work, the effectiveness of information retrieval with a SPARQL example, and the analysis of the characteristics of integrated predicates in the Mid-Ontology. This experimental evaluation is an extension of previous work [28], [29] with more linked data sets to show the adaptability of our method on different combinations of data sets.

3.1 Experimental Data

We used the following four data sets in the LOD cloud to evaluate our approach.

DBpedia is a core cross-domain data set that describes over 3.5 million things including persons, places, music albums, movies, video games, organizations, species, and diseases. DBpedia has more than 232 million RDF triples and more than 8.9 million distinct URIs.

Geonames is a data set that is categorized in the geographic domain and contains more than 7 million unique URIs that represent geographical information on places across the world.

NYTimes data is a small data set that consists of 10,467 subject headings, where 4,978 are about people, 1,489 are about organizations, 1,910 are about locations, and 498 are about descriptors.

LinkedMDB is the Linked Movie DataBase, which contains high-quality interlinks to movie-related data in the LOD cloud as well as links to movie-related web pages. LinkedMDB consists of approximately 6 million RDF triples and more than 0.5 million entities.

Figure 2 shows the SameAs links connecting the above four data sets, plotted using Cytoscape [30]. In this figure, the size of a node is determined by the total number of distinct instances in a data set on a logarithmic scale. The thickness of an arc is determined by the number of SameAs links on a logarithmic scale. The number of SameAs links are labeled on each arc, which links different data sets.

In addition to the SameAs links, there are links at class and property level, which are linked with owl:equivalentClass and owl:equivalentProperty. However, in our experimental data sets, there is no owl:equivalentClass link among them. Although we found 10 pairs of properties linked with owl:equivalentProperty in DBpedia, most of the properties are removed during the data collection phase because of their infrequent usage in the data sets. Hence, these few links between classes and properties do not affect the evaluation results with our approach.

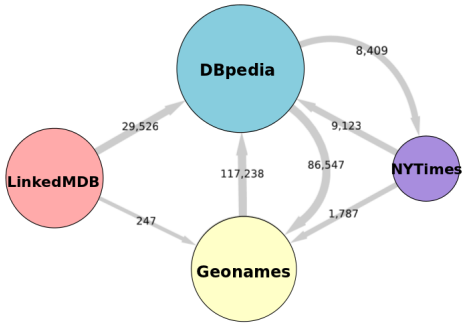

Fig. 2 SameAs links between data sets.

Table 3 Number of instances in each data set.

Data	Number of Instances	*owl:sameAs* Retrieval	Noisy Instances Removal
DBpedia	8,955,728	163,916	115,364
Geonames	7,479,714	128,482	82,055
NYTimes	10,467	10,408	9,242
LinkedMDB	503,242	29,526	28,946

3.2 Evaluation of Data Reduction

We evaluate the effectiveness of data reduction during the data collection phase by comparing the number of distinct instances in the original data sets with the number of distinct instances we extracted after performing the *owl:sameAs* retrieval process and the noisy instances removal process.

DBpedia has served as a hub within the Web of Data because of its breath of topical coverage and the wealth of inward and outward links connecting instances in DBpedia to instances in other data sets [9]. Hence, we select DBpedia as a core data set and collect all the instances that have the SameAs links as do instances in DBpedia.

The instances of DBpedia that contain only the following predicates db-prop:wordnet-type[†], *owl:sameAs*, and db-prop:hasPhotoCollection are defined as "*noisy*" and are removed from the collected linked instances. We reduce the data in such a way because although these instances have SameAs links, from them we cannot learn any information that can represent the characteristics of a DBpedia instance. For example, most of the db-prop:hasPhotoCollection link to a broken link from which we cannot extract information about the link.

The instances of four data sets kept after the data reduction process are shown in Table 3. These retained in-

[†]db-prop: http://dbpedia.org/property/

Table 4 Comparison results of constructed Mid-Ontologies.

Data set	Mid-Ontology	Number of Classes	Number of Predicates	Accuracy	Correctly Labeled	Correctly Grouped	Predicate Richness
DBpedia	MO_1_no_prune_refine	9	228	78.34%	6 (66.67%)	6 (66.67%)	25.33
Geonames	MO_1_no_prune	14	203	82.53%	10 (71.43%)	9 (64.29%)	14.5
NYTimes	MO_1_no_refine	21	166	93.92%	16 (76.19%)	19 (90.48%)	7.90
	MO_1	23	148	94.18%	19 (82.61%)	19 (82.61%)	6.43
DBpedia	MO_2_no_prune_refine	10	240	77.94%	7 (70%)	6 (60%)	24
Geonames	MO_2_no_prune	15	238	78.23%	11 (73.33%)	8 (53.33%)	15.87
LinkedMDB	MO_2_no_refine	19	157	93.16%	15 (78.95%)	17 (89.47%)	8.26
	MO_2	24	150	96.92%	21 (87.5%)	22 (91.67%)	6.25
DBpedia	MO_3_no_prune_refine	9	99	84.76%	7 (77.78%)	6 (66.67%)	11
LinkedMDB	MO_3_no_prune	15	82	85.76%	13 (86.67%)	10 (66.67%)	5.47
NYTimes	MO_3_no_refine	10	120	95.39%	9 (90%)	9 (90%)	12
	MO_3	14	106	93.77%	13 (92.86%)	12 (85.71%)	7.57
DBpedia	MO_4_no_prune_refine	9	169	71.58%	5 (55.56%)	5 (55.56%)	18.78
Geonames	MO_4_no_prune	14	155	79.99%	9 (64.29%)	9 (64.29%)	11.07
NYTimes	MO_4_no_refine	18	111	94.63%	15 (83.33%)	16 (88.89%)	6.17
LinkedMDB	MO_4	22	105	94.84%	19 (86.36%)	20 (90.91%)	4.77

stances are used for our Mid-Ontology learning approach to discover related classes and properties. For example, the db:Hamburg in Table 1 is kept after the data reduction process because it has SameAs links to instances of other data sets and it is not a "noisy" instance. Therefore, we crawl the SameAs links shown in Fig. 2 and collect <predicate, object> pairs from the interlinked instances as listed in Table 1.

Table 3 illustrates the number of distinct instances that exist before and after linked data retrieval and noisy instances removal are performed during the data collection process. The data sets contain 8,955,728 DBpedia instances, 7,479,714 Geonames instances, 10,467 NYTimes instances, and 503,242 LinkedMDB instances. After the linked data retrieval process, we extracted 163,916 DBpedia instances, 128,482 Geonames instances, 10,408 NYTimes instances, and 29,526 LinkedMDB instances, which are 1.83%, 1.72%, 99.44%, and 5.87% of the number of instances in the data sets, respectively.

Then, we pre-process the extracted sub-data set by removing noisy DBpedia instances. After the noisy instances removal, we obtained 115,364 DBpedia instances, 82,055 Geonames instances, 9,242 NYTimes instances, and 28,946 LinkedMDB instances, which are 70.78%, 63.86%, 88.80%, and 98.04% of the number of instances in the extracted linked data, respectively.

We dramatically scaled down the data sets by collecting information of linked instances in the data collection phase so as to keep instances that share related information. Furthermore, we successfully removed noisy instances, which may affect the quality of the constructed ontology.

3.3 Ontology Evaluation

To evaluate the quality of the constructed Mid-Ontology (MO), we calculate the accuracy of the Mid-Ontology using the following formula:

$$ACC(MO) = \frac{\sum_{i=1}^{n} \frac{|Correct\ Predicates\ in\ C_i|}{|C_i|}}{n}$$

where n is the number of classes in the Mid-Ontology, and $|C_i|$ indicates the number of predicates in class C_i. The *ACC(MO)* is the average of the accuracy of each class in the Mid-Ontology. If all related or identical predicates are correctly integrated in each class, *ACC(MO)* reaches 1.

Table 4 shows the improvements achieved by our Mid-Ontology approach through a comparison of the Mid-Ontologies constructed with and without our approach using different combinations of data sets, as illustrated in the first column. The main features of our approach are the *PO* set pruning with similarity measures and the *PO* set refining by checking the ranges and domains of predicates. The second column lists the Mid-Ontologies constructed by different approaches, that is, MO_i_no_prune_refine is constructed without the pruning and refining processes, MO_i_no_prune is constructed without the pruning process, MO_i_no_refine is constructed without the refining process, and MO_i is constructed with both the pruning and refining processes. The third column lists the number of classes, and the fourth column lists the number of predicates in the constructed Mid-Ontology. The following columns list the accuracy, the percentage of correctly labeled terms, the percentage of correctly grouped classes, and the Predicate Richness (PR) of the constructed Mid-Ontology.

3.3.1 Evaluation of Pruning Process

To evaluate the accuracies of the constructed Mid-Ontologies, we manually check predicates in each class to determine whether they share identical or related information, and we examine whether each term can represent the predicates in that class without disambiguation. The performance of the pruning process can be evaluated by comparing the results of MO_i_no_prune_refine and MO_i_no_refine, and the results of MO_i_no_prune and MO_i.

For example, in the first case, which is performed on DBpedia, Geonames, NYTimes, the pruning process significantly improved the accuracy of the Mid-Ontology, that is, from 78.34% to 93.92% and from 82.53% to 94.18%, with a p-value 0.01. Here, the p-value is calculated using a t-test

to measure the statistical significance of improvement. The p-value calculated on eight pairwise accuracies from four different cases is 2.0E-5, indicating that our pruning process significantly improves the accuracy of the Mid-Ontology.

The pruning process is applicable to any type of objects, but it is not an optimized method. Since we consider the numerical values of objects as literals in the similarity matching, it may cause incorrect similarity values. This problem can be solved by applying different similarity measurements for different types of objects as introduced in [31].

3.3.2 Evaluation of Refining Process

We can compare the results of MO_i_no_prune_refine and MO_i_no_prune, and the results of MO_i_no_refine and MO_i to evaluate the performance of the refining process. For example, in the first case, the refining process improved the accuracy of the Mid-Ontology, that is, from 78.34% to 82.53% and from 93.92% to 94.18%. Although the improvements of the refining process from four different cases are not significant, with a p-value 0.056, this value is close to the significant level 0.05.

The reason of insignificant improvements on the refining process is that most of the predicates in the ontologies have no definition of range and domain. For instance, there is no domain and range information in the LinkedMDB and NYTimes data sets. In Geonames, there are 12 range information and 21 domain information. However, all the ontology predicates are defined as geo:Feature. DBpedia has over 6000 predicates for describing instances, but less than 20% of them have domain and range information. We will seek for a solution to improve the performance of refining process on the data sets, which are lack of domain and range information.

The decrease of the accuracy in this experiment is caused in the third case, where the accuracy of MO_3 is slightly lower than that of MO_3_no_refine. One reason for this decrease is that in the MO_3, db-onto:producer and db-prop:playername are divided from the largest class mid-onto:name and grouped into a new class mo-onto:playername after the refinement step because both predicates indicate the name of a person. However, the db-onto:producer is related to movies and db-prop:palyername is related to sports. Hence, we define this set as incorrect, and the accuracy of mo-onto:playername is 50%, decreasing the accuracy of MO_3 by approximately 2.27%. Another reason for the accuracy decrease is that there are not as many links as there are in other cases; in other words, fewer DBpedia instances are retrieved during the data collection. The number of DBpedia instances in the third case is only 30% of the number of instances in the last case. Hence, because of the limited number of extracted instances, the refining process failed to improve the accuracy.

3.3.3 Evaluation of Correctly Lableled and Grouped Sets

As we can see from Table 4, the percentages of correctly labeled sets and correctly grouped classes of MO_i are improved compared to the results obtained when the group pruning or group refining processes were not performed.

Although the percentage of correctly labeled terms for MO_i is higher than in the case of methods that do not involve group pruning or group refining, such as MO_i_no_prune_refine, MO_i_no_prune, and MO_i_no_refine, the label accuracy is not affected by the pruning and refining processes. Some of the automatically labeled terms are ambiguous to represent the precise meaning of the terms in a set. Hence, we define these unclear terms as incorrectly labeled terms. An example of an incorrectly labeled term in the fourth case in Table 4 is "date", which should represent the "release date" of a movie. The term "date" is selected because it appeared most frequently in the preprocessed terms of the set. However, this term is too ambiguous that we can not figure out whether it is a release date of a movie.

The percentage of correct groups is significantly improved after performing pruning process, with a p-value 6.27E-6. However, the improvement of the refining process is not significant, with a p-value 0.3. Although the accuracies of correct groups are high with the experimental data sets, there are some incorrect groups. An example of an incorrect group is the one including predicates about longitude and latitude. This wrong group is caused by the coincidence when the value of longitude and latitude are similar. Furthermore, due to the lack of domain and range information of the predicates that indicating longitude and latitude, they are not separated during the refining process.

3.3.4 Evaluation of Predicate Richness

As Table 4 shows, when both pruning and refining are conducted, the total number of predicates from the data sets are decreased and the total number of classes are increased. To determine whether we successfully retrieved related predicates and removed redundant predicates, we introduce the term Predicate Richness (PR), which is calculated using the following formula:

$$PR = \frac{|Number\ of\ Predicates|}{|Number\ of\ Classes|}$$

where PR indicates the average number of predicates in a class of the ontology. As we can see from Table 4, the cases with a low PR have a high accuracy. Hence, according to the high accuracy and the low PR, we can conclude that through the pruning and refining process we successfully removed redundant predicates, which may reduce the accuracy of the Mid-Ontology.

3.4 Comparison Results with Previous Work

We compare our experimental results performed on DBpe-

Table 5 Comparison results on DBpedia, Geonames, and NYTimes.

Mid-Ontology	Number of Classes	Number of Predicates	Accuracy	Correctly Grouped
Previous MO	29	180	90.10%	22 (75.86%)
Current MO	23	148	94.18%	19 (82.61%)

Table 6 Predicates grouped in mo-onto:date.

```
<rdf:Description rdf:about="mo-onto:date">
    <mo-prop:hasMembers rdf:resource="db-prop:released"/>
    <mo-prop:hasMembers rdf:resource="db-onto:releaseDate"/>
    <mo-prop:hasMembers rdf:resource="mdb-movie:initial_release_date"/>
    <mo-prop:hasMembers rdf:resource="dc:date"/>
</rdf:Description>
```

dia, Geonames, and NYTimes with previous work introduced in [28]. The main difference in our current work is that we did not consider instances in the core data set if there is no link to instances from other data sets. Furthermore, during the PO collection, we also collect from redirected instances which may contain more information than the original instance.

The comparison results of the Mid-Ontology between previous work and current work are described in Table 5. The number of classes and predicates are decreased in our current work, because we did not consider instances which have SameAs links, but only link to other instances in the same data set. The accuracy and correctly grouped predicates are improved by 4.5% and 8.9%, respectively in our current work comparing with previous work. Since we collected more information from redirected instances, which usually contain more PO pairs than the original instance, the accuracy of integrated predicates is increased comparing with previous work.

3.5 Evaluation with a SPARQL Example

We evaluate the effectiveness of information retrieval with the Mid-Ontology constructed via our approach using DBpedia, Geonames, NYTimes, and LinkedMDB data sets by presenting a SPARQL query example.

Table 6 shows one of the classes in the Mid-Ontology, that integrates predicates indicating the release date of a movie from DBpedia and LinkedMDB. The predicates db-prop:released and db-onto:releaseDate are used in DBpedia, while mdb-movie:initial_release_date and dc:date[†] are used in LinkedMDB. This set does not contain any NYTimes and Geonames predicates because there is no predicate that indicates a movie's release date in both data sets. Table 7 shows a SPARQL example in which this mo-onto:date is used to find movies that are released on 2000-10-03. This SPARQL query automatically queries with all the predicates listed under mo-onto:date, as shown in Table 6.

We identify 46 movies with mo-onto:date, where 44 are from DBpedia and 2 are from LinkedMDB. However, with the single predicate listed in Table 8, we can find 37, 43, 2, and 2 movies. Because the predicates grouped in this class all correctly represent the release date of a movie, the returned results are all correct. The result queried with mo-onto:date is a combination of the results retrieved with each predicate in that set. Furthermore, it is difficult to manually find all four predicates that indicate the release date in different data sets.

Table 7 A SPARQL example: Find movies released on 2000-10-03.

```
SELECT DISTINCT ?movies
        WHERE{ <mo-onto:date> mo-prop:hasMembers ?prop.
        ?movies ?prop ?date.
         FILTER REGEX(?date, "2000-10-03").}
```

Table 8 SPARQL example results with each single predicate.

Single property for the release date of a movie	Number of Results
db-prop:released	37
db-onto:releaseDate	43
mdb-movie:initial_release_date	2
dc:date	2

Table 9 Sample classes in the Mid-Ontology.

Data sets	Mid-Ontology class
DBpedia	mo-onto:producer
	mo-onto:areacode
DBpedia & Geonames	mo-onto:population
	mo-onto:postal
DBpedia & LinkedMDB	mo-onto:date
DBpedia & Geonames & NYTimes & LinkedMDB	mo-onto:name

As this example shows, our approach simplifies SPARQL queries and returns all the possible results without user interaction; conversely, it is time-consuming to find each single predicate manually through user interaction.

3.6 Characteristics of Integrated Ontology Predicates

We evaluate whether our approach successfully integrates related predicates by illustrating examples of classes in the Mid-Ontology. Table 9 shows some of the classes in the Mid-Ontology, which integrated predicates from the DBpedia, Geonames, NYTimes, and LinkedMDB instances.

The classes listed in the first row include only predicates from DBpedia ontology, which indicate the producer of a movie and the area code of a place. The second row lists classes that integrate predicates from both DBpedia and Geonames, which indicate the population and postal code of a place. The third row includes the release date of a movie, which integrates predicates from DBpedia and LinkedMDB. The last row includes a predicate that integrates predicates from DBpedia, Geonames, NYTimes, and LinkedMDB, which indicates the name of persons, places, news, or movies.

From the characteristics of the integrated classes, we can observe that the linked instances between DBpedia and Geonames are about places, and the instances that link DBpedia and LinkedMDB are based on the release date of a movie, and all the predicates that refer to the label of a thing in four data sets are integrated in mo-onto:name. The predicate mo-onto:name includes the name of a place, a movie,

[†]dc: http://purl.org/dc/terms/

an actor or actress, a news title, etc.

However, we may miss some predicates that should be integrated together. For example, the predicate mdb-movie:runtime[†] and db-prop:runtime both indicate the runtime of a movie, but these two predicates do not appear in the final Mid-Ontology. This failure of integration occurs because we filter out sets that have a frequency lower than predefined threshold: 0.1% of the number of retrieved core data instances. Hence, when the threshold is high, some sets such as "runtime" and "prominence" are filtered out.

Another interesting observation is that some predicate terms that are written in different languages are integrated together. For instance, db-prop:einwohner is integrated in the mo-onto:population, which means "the population" in German. Other predicates such as geo-onto:population and db-onto:populationTotal have clear terms in English, which are also integrated in the mo-onto:population. Another example is db-prop:vorwahl which is also in German meaning the area code of a place. Two predicates db-prop:areaCode and db-onto:areaCode are integrated together with db-prop:vorwahl in the mo-onto:areaCode. The terms written in English are easy to understand, but it is impossible to identify the term "einwohner" or "vorwahl" if we do not know German.

4. Discussion

Experimental results demonstrate that our Mid-Ontology learning approach successfully integrates predicates from different data sets. The automatically constructed Mid-Ontology has a high quality and can be applied in the information retrieval field. Because the Mid-Ontology integrates the most related predicates, we can search related triples or instances from the LOD cloud with a simple SPARQL query.

Furthermore, our Mid-Ontology learning approach is implemented with the collected data set that is extracted with *owl:sameAs*. Hence, with our Mid-Ontology in a SPARQL query, it is possible to find missing links that should be linked with *owl:sameAs*. For example, the predicate mo-onto:date has predicates in DBpedia and LinkedMDB that indicate the release date of a movie. Therefore, we can find the same movie in DBpedia and LinkedMDB by searching movies that are released on the same date with the same title. From the results of the SPAQL query in Table 7, we found one missing link that should connect two instances db:Scooby-Doo_and_the_Alien_Invaders and mdb-film:45394[††]. The db:Scooby-Doo_and_the_Alien_Invaders has predicates db-prop:released and db-onto:releaseDate with the value "2000-10-03", and the LinkedMDB instance mdb-film:45394 has predicates mdb-movie:initial_release_date and dc:date with the same value, "2000-10-03". This LinkedMDB instance also indicates the movie "Scooby-Doo and the Alien Invaders", but there is no *owl:sameAs* link between these two instances.

Therefore, we can find missing links with our Mid-Ontology if there exist predicates from different domains grouped under the same Mid-Ontology class. In our constructed Mid-Ontology, we can find missing links according to the predicates mo-onto:birthdate, mo-onto:population, mo-onto:postalcode, etc. In some SPARQL queries, we can find many missing links that should be connected with "owl:sameAs". For example, we can use mo-onto:birthdate to find out persons who were born on the same day. With mo-onto:birthdate, we found 18 instances of persons who were born on "1961-08-04", where 4 of them indicate "Barack Obama", but no SameAs links among these instances. Hence, we can add SameAs links among the instances.

Although, it is difficult to find out all the missing links in the linked data, we can discover missing links with a specific SPARQL query template.

5. Related Work

Some researchers have proposed similar ideas about constructing an intermediate-layer ontology and reducing data sets. For instance, the authors in [32] automatically generated alignments from linked instances in the Linked Data, especially geospatial, zoology, and genetics data sources. During data preprocessing, they only considered linked instances and removed unimportant properties to reduce the search space. Their algorithm discovers equivalent and subsumption relations and models one Linked Data source through ontology alignment. However, their approach is limited to specific domains of data sets, while our approach can be applied to data sets from any domain. Furthermore, they only focused on ontology alignment at the class level and did not consider at the property level, which is useful to find related instances from different data sets.

Some researchers have proposed the construction of an intermediate-level ontology to connect general ontologies and an upper ontology. The authors in [33] introduced a method to construct an intermediate-level ontology by mapping an upper ontology, PROTON, to the concepts of DBpedia, Geonames, and Freebase described in the FactForge. They enriched the upper ontology by adding 166 new classes and 73 new properties; the resulting ontology was large. The Semantic Web users would have to understand this large ontology to construct SPARQL queries. In order to link the concepts and properties in DBpedia and Geonames with PROTON, they manually designed the rules with subsumption relations from FactForge to PROTON. This approach is not scalable when there are many data sets to be linked with the PROTON. In contrast, our approach is fully automated to integrate related predicates from linked instances.

The authors in [34] applied debugging method for mapping lightweight ontologies. They applied machine learning method to learn a classifier that can determine disjointness of two classes, with the features of the taxonomic overlap,

[†]mdb-movie: http://data.linkedmdb.org/resource/movie/

[††]mdb-film: http://data.linkedmdb.org/resource/film/

semantic distance, object properties, label similarity, and WordNet similarity. Although the experimental results show improvements comparing with other ontology matching systems, their method is limited to expressive lightweight ontologies and manual preprocessing is necessary. In fact, real data sets are lack of expressive disjointness axioms and contain large ontologies. They evaluated mapping results with gold standard that is created by three people who are familiar with the ontology in the specific domain. However, this kind of measurement is infeasible for large ontologies from various domains. Since we use real LOD data sets with large ontologies, we are not able to create gold standard ontology mappings created by experts to evaluate our ontology.

The scalability and time efficiency is important for analyzing large data. The authors in [35] introduced a time-efficient approach for large-scale link discovery in the linked data sets. They utilized the triangle inequality to compute the similarities between instances and filter out a large number of instance pairs to reduce the computation space. Although they significantly reduced the running time in the experiments, they did not evaluated the quality of the discovered links. Furthermore, the best number of exemplars and the threshold for different pairs of source and target data sets should be tested, and it is time-consuming to discover links on several large data sets.

The authors in [36] analyzed the basic properties of the SameAs network, the Pay-Level-Domain network, and the Class-Level similarity network. They analyzed the Pay-Level-Domain network to examine how data publishers are connected by comparing the five most frequent types. However, when only frequent types are considered, it is not possible to determine exactly how data are connected.

In contrast to the approaches adopted in the related research described above, our Mid-Ontology learning approach is aimed at constructing a small ontology by integrating predicates and can be directly applied to Semantic Web applications. With our Mid-Ontology, we can easily determine the types of things that are linked together by observing the characteristics of the integrated predicates. Furthermore, user interaction is not needed for the Mid-Ontology construction, and our approach is applicable to linked data sets as long as they are connected by SameAs links.

The drawbacks of our approach are that a hub data set is necessary to extract linked instances and that related predicates cannot be identified if data sets are not directly connected in the LOD cloud. One possible solution is to investigate the connected components [36] in the LOD cloud by applying clustering technology and by analyzing the contents of the connected components.

6. Conclusion and Future Work

In this paper, we proposed a Mid-Ontology learning approach that involves the use of Linked Open Data and can help Semantic Web application developers integrate diverse ontology schema without learning the entire ontology schema. The main procedures of our approach are data collection, the ontology predicate grouping process, and Mid-Ontology construction. The predicate grouping algorithm applied lexical similarity matching to collect similar predicates and implemented the relation extraction method to refine predicate sets. Our approach can automatically extract the most related predicates between linked data sets and integrate them in the Mid-Ontology. Experiments show that the amount of data can be dramatically reduced in the data collection phase and that the accuracy of the Mid-Ontology can be significantly improved by pruning and refining processes with different combinations of data sets. Comparing with our previous research [28], the accuracy is improved when performing on the same data sets. Furthermore, with the Mid-Ontology, related information can be effectively retrieved by a simple SPARQL query, and we can easily recognize how the instances are linked from the characteristics of integrated predicates. The Mid-Ontology constructed with our approach can be used to search semantic data or identify missing SameAs links.

In future work, we will apply our approach to the Billion Triple Challenge (BTC) data set, which is collected by crawling real-world linked data. Because our current approach only considers data sets directly linked with a hub data set, it cannot extract relations by crawling links at more than one depth. We plan to extend our approach so that it can crawl at two or three depths of links to collect interesting information from the linked data.

References

[1] G. Klyne and J.J. Carroll, "Resource description framework (RDF): Concepts and abstract syntax," W3C Recommendation, 2004. http://www.w3.org/TR/rdf-concepts/
[2] T. Berners-Lee, "Linked data - Design issues," 2006. http://www.w3.org/DesignIssues/LinkedData.html
[3] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein, "OWL Web ontology language reference," W3C Recommendation, 2004. http://www.w3.org/TR/owl-ref/
[4] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, and S. Decker, "Sig.ma: Live views on the web of data," J. Web Semantics, vol.8, no.4, pp.355–364, 2010.
[5] T. Finin, Y. Peng, R. Scott, C. Joel, S.A. Joshi, P. Reddivari, R. Pan, V. Doshi, and L. Ding, "Swoogle: A search and metadata engine for the semantic web," Proc. Thirteenth ACM Conference on Information and Knowledge Management, pp.652–659, 2004.
[6] G. Kobilarov, T. Scott, Y. Raimond, S. Oliver, C. Sizemore, M. Smethurst, C. Bizer, and R. Lee, "Media meets semantic web — how the bbc uses dbpedia and linked data to make connections," Proc. Sixth European Semantic Web Conference, pp.723–737, 2009.
[7] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data - The story so far," Int. J. Semantic Web and Information Systems, vol.5, no.3, pp.1–22, 2009.
[8] S. Auer and J. Lehmann, "Creating knowledge out of interlinked data," Semantic Web J., vol.1, no.1-2, pp.97–104, 2010.
[9] T. Heath and C. Bizer, Linked Data: Evolving the Web into a Global Data Space, Morgan & Claypool, 2011.
[10] E. Prudhommeaux and A. Seaborne, SPARQL query language for RDF. W3C Recommendation, 2008. http://www.w3.org/TR/rdf-sparql-query/
[11] S. Pavel and J. Euzenat, "Ontology matching: State of the art and future challenges," IEEE Trans. Knowl. Data Eng., vol.99,

no.PrePrints, 2011.
[12] L. Drumond and R. Girardi, "A survey of ontology learning procedures," Proc. Third Workshop on Ontologies and their Applications, CEUR Workshop Proceedings, vol.427, 2008.
[13] L. Zhou, "Ontology learning: State of the art and open issues," Information Technology and Management, vol.8, pp.241–252, 2007.
[14] P. Buitelaar and B. Magnini, "Ontology learning from text: An overview," in Ontology Learning from Text: Methods, Applications and Evaluation, IOS Press, pp.3–12, 2005.
[15] J.A. Gulla, H.O. Borch, and J.E. Ingvaldsen, "Ontology learning for search applications," Proc. Sixth International Conference on Ontologies, DataBases, and Applications of Semantics, pp.1050–1062, 2007.
[16] N. Choi, I.Y. Song, and H. Han, "A survey on ontology mapping," ACM SIGMOD Record, vol.35, pp.34–41, 2006.
[17] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives, "Dbpedia: A nucleus for a web of open data," Proc. Sixth International Semantic Web Conference, pp.11–15, 2007.
[18] H. Halpin, P.J. Hayes, J.P. McCusker, D.L. McGuinness, and H.S. Thompson, "When owl: sameas isn't the same: An analysis of identity in linked data," Proc. Ninth International Semantic Web Conference, pp.305–320, 2010.
[19] J. Euzenat and P. Shvaiko, Ontology Matching, Springer-Verlag, Heidelberg, 2007.
[20] R. Ichise, "Machine learning approach for ontology mapping using multiple concept similarity measures," Proc. Seventh IEEE/ACIS International Conference on Computer and Information Science, pp.340–346, 2008.
[21] R. Ichise, "An analysis of multiple similarity measures for ontology mapping problem," Int. J. Semantic Comput., vol.4, no.1, pp.103–122, 2010.
[22] T. Pedersen, S. Patwardhan, and J. Michelizzi, "Wordnet::similarity: Measuring the relatedness of concepts," Proc. Nineteenth National Conference on Artificial Intelligence, pp.1024–1025, 2004.
[23] C. Fellbaum, ed., WordNet: An Electronic Lexical Database, MIT Press, 1998.
[24] M.F. Porter, "An algorithm for suffix stripping," in Readings in Information Retrieval, pp.313–316, Morgan Kaufmann Publishers, 1997.
[25] P. Cimiano, Ontology Learning and Population from Text: Algorithms, Evaluation and Applications, Springer-Verlag New York, 2006.
[26] D. Brickley and R. Guha, "RDF vocabulary description language 1.0: RDF schema," W3C Recommendation, 2004. http://www.w3.org/TR/rdf-schema/
[27] O. Erling and I. Mikhailov, Rdf support in the virtuoso dbms, Networked Knowledge-Networked Media, Springer, pp.7–24, 2009.
[28] L. Zhao and R. Ichise, "Mid-ontology learning from linked data," Proc. Joint International Semantic Technology Conference, 2011.
[29] L. Zhao and R. Ichise, "One simple ontology for linked data sets," Proc. Tenth International Semantic Web Conference: Posters and Demos, 2011.
[30] P. Shannon, A. Markiel, O. Ozier, N.S. Baliga, J.T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: A software environment for integrated models of biomolecular interaction networks," Genome Res, vol.13, no.11, pp.2498–2504, 2003.
[31] L. Zhao and R. Ichise, "Graph-based ontology analysis in the linked open data," Proc. Eighth International Conference on Semantic Systems, pp.56–63, 2012.
[32] R. Parundekar, C.A. Knoblock, and J.L. Ambite, "Linking and building ontologies of linked data," Proc. Ninth International Semantic Web Conference, pp.598–614, 2010.
[33] M. Damova, A. Kiryakov, K. Simov, and S. Petrov, "Mapping the central lod ontologies to proton upper-level ontology," Proc. Fifth International Workshop on Ontology Matching, pp.61–72, 2010.
[34] C. Meilicke, J. Völker, and H. Stuckenschmidt, "Learning disjointness for debugging mappings between lightweight ontologies," Proc. Sixteenth International Conference on Knowledge Engineering and Knowledge Management, pp.93–108, 2008.
[35] A.C.N. Ngomo and S. Auer, "Limes-a time-efficient approach for large-scale link discovery on the web of data," Proc. Twenty-Second International Joint Conference on Artificial Intelligence, pp.2312–2317, 2011.
[36] L. Ding, J. Shinavier, Z. Shangguan, and D.L. McGuinness, "Sameas networks and beyond: Analyzing deployment status and implications of owl: sameas in linked data," Proc. Ninth International Semantic Web Conference, pp.145–160, 2010.

Lihua Zhao received her BSc degree in software engineering from Jilin University, China, in 2007 and her MSc degree in machine learning and data mining from the University of Bristol, U.K., in 2009. She is currently a Ph.D. candidate at the National Institute of Informatics in Japan. Her research interests include Semantic Web and data mining.

Ryutaro Ichise received his Ph.D. in computer science from the Tokyo Institute of Technology, Tokyo, Japan, in 2000. From 2001 to 2002, he was a visiting scholar at Stanford University. He is currently an associate professor in principles informatics research division at the National Institute of Informatics in Japan. His research interests include machine learning, Semantic Web, and data mining.