3C1-OS-13a-4

Integrating Heterogeneous Ontology Schema from LOD

Lihua Zhao Ryutaro Ichise

National Institute of Informatics, Tokyo, Japan

The Linked Open Data (LOD) includes over 31 billion Resource Description Framework (RDF) triples interlinked by around 504 million RDF links (as of September 2011). Linking related instances in the LOD can help Semantic Web application developers easily retrieve information from various data sets. However, because of the heterogeneity of ontology schema in the LOD, it is difficult for them to query the data sets without manually learning ontologies. An ontology integration method can help us to detect and integrate important ontology schema for linking related data. Since the links between related resources construct a linked SameAs Graph, we can detect graph patterns from the linked data. We can discover related classes and properties that are used for linking related instances by analyzing the SameAs graph patterns. We apply ontology similarity matching on the graph patterns to identify related predicates from different ontology schema. Using the automatically integrated ontology schema, Semantic Web application developers can easily understand the ontologies and effectively query on the linked data sets.

1. Introduction

The Linked Open Data (LOD) is a collection of machine-readable structured data connected by *owl:sameAs*, which refers to related or identical instances in diverse data [Bizer 09]. Although a huge amount of data sets are published in the LOD cloud, there is no standard ontology for all the data sets, but all kinds of ontologies which cause the ontology heterogeneity problem. A commonly used method to overcome the ontology heterogeneity problem is the ontology matching, which finds corresponding mappings between ontologies [Pavel 11]. Since it is time-consuming and infeasible to manually inspect large ontologies of linked data, we need an automatic or semi-automatic method to integrate heterogeneous ontologies.

In this paper, we propose a semi-automatic approach to integrate heterogeneous ontologies by analyzing the SameAs Graphs at both class and property level. From the SameAs Graphs, we retrieve graph patterns and analyze linked instances to integrate related classes and properties for the whole data sets and construct an integrated ontology. Experimental results show that our approach effectively integrates core classes and properties from linked instances with minor manual revision. With the integrated ontology, we can detect mistaken usage of properties and can suggest a proper class description for an instance.

2. Our Approach

Figure 1 shows the architecture of our approach, which consists of graph pattern extraction, <Predicate, Object> collection, related classes and properties grouping, integration of classes and properties for all graph patterns, and manual revision of the integrated ontology.

2.1 Graph Pattern Extraction

The instances which refer to the same thing are interlinked by *owl:sameAs* in the LOD cloud. In order to investigate on the linked instances, we collect all the instances that have *owl:sameAs* (SameAs) link to other instances.

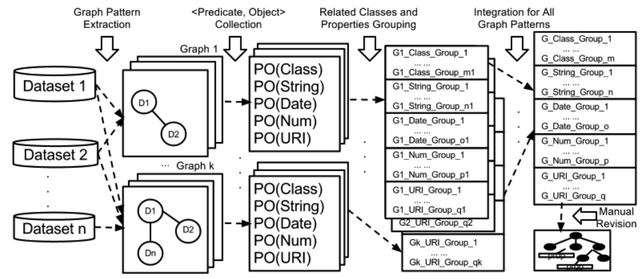

Figure 1: Architecture of our approach.

An undirected **SameAs Graph** SG = (V, E, L), where V is a finite set of vertices, $E \subseteq V \times V$ is a set of edges, and L is a set of labels of V. Here, a vertex represents the URI of an instance and a label is based on the pay-level domain of the URI. We extract graph patterns from the SameAs Graphs and analyze the instances to retrieve related classes and properties.

2.2 <Predicate, Object> Collection

An instance is described by a collection of RDF triples in the format of <subject, predicate, object>, where the subject refers to the URI of an instance. Since a SameAs Graph contains linked instances, we collect all the <Predicate, Object> (PO) pairs of the linked instances as the content of the SameAs Graph.

We classify the objects of PO pairs into five different types: Class, String, Date, Number, and URI. The Class of a URI is defined by rdf:type[*1] and skos:inScheme[*2]. The other four types can be identified by the built-in data types, which are followed by the symbol "^^". If the data types are not given expressively in the RDF triples, we classify the value of an object as Number if it only contains numbers, as URI if it starts with "http://", and as String otherwise.

*1 rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
*2 skos: http://www.w3.org/2004/02/skos/core#

Contact: 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, lihua@nii.ac.jp, ichise@nii.ac.jp

2.3 Related Classes and Properties Grouping

The classes of ontologies have subsumption relations such as owl:subClassOf and skos:inScheme. The triple $< C_1$ owl:subClassOf $C_2 >$ or $< C_1$ skos:inScheme $C_2 >$ means the concept of class C_1 is more specific than the concept of class C_2. In order to identify the types of linked instances, we focus on the most specific classes from each SameAs Graph by tracking the owl:subClassOf and skos:inScheme.

We perform ontology similarity matching method on the collected PO pairs to find out related properties (predicate in RDF triple). We apply exact matching and similarity matching to extract related predicates, and refine the groups of predicates with extracted relations as introduced in [Zhao 11]. All the PO pairs with the same predicate or object are grouped in a set S as an initial set of predicates.

We adopted three string-based similarity measures: JaroWinkler distance, Levenshtein distance, and n-gram [Ichise 10]. String-based similarity measures are applied to compare objects which are classified as String. The similarity of objects are calculated as follows:

$$Sim(O_{S_i}, O_{S_j}) \begin{cases} 1 - \frac{|O_{S_i} - O_{S_j}|}{O_{S_i} + O_{S_j}} & \text{if } O \text{ is Number} \\ \frac{StrSim(O_{S_i}, O_{S_j})}{3} & \text{if } O \text{ is String} \end{cases} \quad (1)$$

where $StrSim(O_{S_i}, O_{S_j})$ is the average of the three string-based similarity values and the term O_S indicates the objects stored in S. $SetSim(S_i, S_j)$ is the similarity between two sets S_i and S_j, which is calculated using the formula:

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

where the term T_S indicates the pre-processed terms of the predicates in S and $WNSim(T_{S_i}, T_{S_j})$ is the average of the nine applied WordNet-based similarity values [Zhao 11]. If the $SetSim(S_i, S_j)$ is higher than a predefined similarity threshold, we merge S_i and S_j.

2.4 Integration for All Graph Patterns

For each SameAs graph pattern, we automatically extract integrated groups of classes and properties which are classified into String, Date, Number, and URI. Then we construct an ontology based on the integrated groups of classes and properties with automatically selected concept of each group and designed relations. The ex-onto:Term is designed to represent a class, and the ex-prop:term is designed to represent a property. We designed a predicate ex-prop:hasMemberClasses to link a set of classes with an integrated class ex-onto:Term, and designed ex-prop:hasMemberDataTypes to link a set of properties with an integrated property ex-prop:term.

2.5 Manual Revision

The automatically constructed ontology integrates classes and properties from different data sets. However, not all the terms of classes and properties are properly selected and some groups of properties lack of rdfs:domain information. Hence, we need experts to work on revising the integrated ontology by choosing proper terms, by adding domain information for properties, and by amending groups of classes and properties.

3. Experiments

We retrieved 13 graph patterns from the SameAs Graphs with the data sets LinkedMDB (M), DBpedia (D), NYTimes (N), and Geonames (G). By analyzing the integrated classes from the 13 graph patterns, we found what kind of instances are interlinked from different data sets. For example, the SameAs Graphs including instances from (M, D, N, G) or (M, D, G) are about country, the SameAs Graphs with (M, D, N) are about actor, the SameAs Graphs with (D, N, G) are about place, the SameAs Graphs with (M, D) are about film and actor, the SameAs Graphs with (D, G) are about place and organization, and the SameAs Graphs with (D, N) are about person, organization, and place.

Furthermore, we can identify different descriptions of the classes, for example, the predicates db-onto:Country[*3], geo-onto:A.PCLI[*4], and mdb:country[*5] indicate country in D, G, and M, respectively. An example of integrated properties is the population which includes mdb:country_population, geo-onto:population, db-onto:populationTotal, etc.

The integrated ontology includes 48 groups of classes and 38 groups of properties after minor manual revision. We can effectively query on various data sets and detect mistakenly used properties in the real data sets with the integrated ontology.

4. Conclusion

In this paper, we proposed a semi-automatic ontology integration method to solve the ontology heterogeneity problem in the LOD. Our approach apply ontology similarity matching on the graph patterns extracted from the linked instances, and analyze ontologies at both class and property level. Experimental results show that we successfully discovered related classes and properties which are important to link the instances.

References

[Bizer 09] Christian Bizer, Tom Heath and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[Pavel 11] Shvaiko Pavel and Jérôme Euzenat. Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints), 2011.

[Zhao 11] Lihua Zhao and Ryutaro Ichise. Mid-Ontology Learning from Linked Data. In *Proceedings of the 1st Joint International Semantic Technology Conference*, 2011.

[Ichise 10] Rutaro Ichise. An Analysis of Multiple Similarity Measures for Ontology Mapping Problem. *International Journal of Semantic Computing*, 4(1):103–122, 2010.

*3 db-onto: http://dbpedia.org/ontology/
*4 geo-onto: http://www.geonames.org/ontology#
*5 mdb: http://data.linkedmdb.org/resource/movie/