Generation and Matching of Ontology Data for the Semantic Web in a Peer-to-peer Framework

Chao Wang, Jie Lu and Guangquan Zhang

Faculty of Information Technology, University of Technology, Sydney
PO Box 123, Broadway, NSW 2007, Australia
{cwang, jielu, zhangg}@it.uts.edu.au

Abstract. The abundance of ontology data is very crucial to the emerging semantic web. This paper proposes a framework that supports the generation of ontology data in a peer-to-peer environment. It not only enables users to convert existing structured data to ontology data aligned with given ontology schemas, but also allows them to publish new ontology data with ease. Besides ontology data generation, the common issue of data overlapping over the peers is addressed by the process of ontology data matching in the framework. This process helps turn the implicitly related data among the peers caused by overlapping into explicitly interlinked ontology data, which increases the overall quality of the ontology data. To improve matching accuracy, we explore ontology related features in the matching process. Experiments show that adding these features achieves better overall performance than using traditional features only.

1 Introduction

Ontology has been realized to be an essential layer of the emerging semantic web [1]. According to description logics (DL) [2], an ontology as a knowledge base normally consists of a "TBox" and a "ABox". The TBox consists of concepts and their relations while the ABox consists of instances of concepts or individuals. For an ontology expressed in the web ontology language (OWL) [3], we use the term *ontology schema* and *ontology data* for the part corresponding to the notion of TBox and ABox respectively. Through the semantic web search engine Swoogle (http://swoogle.umbc.edu), we've found that there are plenty of ontology schemas available over the web. But in contrast, ontology data does not seem to be very abundant.

The abundance of information and data in the current web has made the web an important place for people to seek information. This leads us to believe that the abundance of ontology data in the semantic web is very crucial. Therefore, besides the development of ontology schemas, the generation of ontology data also plays an important role for the semantic web.

Generation of ontology data can be achieved through different ways. This paper proposes a framework that supports the generation of ontology data through conversion and authoring. It is designed with the peer-to-peer architecture, which is more flexible and scalable in terms of ontology data management and distribution. Although several (ontology) data management or sharing frameworks based on peer-to-peer architecture have been proposed (e.g. Edutella [4], Piazza [5],and etc), our framework has an advantage that differentiate it from them in that it deals with the issue of ontology data matching across peers. Normally, the ontology data for a certain domain contributed by one peer may not be complete but may be implicitly related to those contributed by other peers as data overlapping often occurs. Our framework, designed with a matching process to discover the implicit relations, is able to help reduce the redundancy and increase the amount of richly inter-linked ontology data.

The rest of the paper is organized as follows. Section 2 discusses related work. An overview of the framework is given in Section 3 . Section 4 describes how the framework supports the generation of ontology data. Section 5 presents the function of ontology data matching. Experimental results about ontology data matching are shown in Section 6. Section 7 concludes the paper.

2 Related Work

We discuss relate work from two aspects: ontology data generation and data matching.

There are a few ways to generate ontology data. Some ontology development tools such as Pretege (http://protege.stanford.edu), SWOOP [6] can be used to create new ontology data as well as to develop ontology schemas. However, as these tools are designed with an emphasis on knowledge representation, they are often used by experienced domain experts familiar with ontology related techniques. Ordinary users may have difficulty using it, which hinders the generation of ontology data in a large scale. Annotation of existing data with ontologies is another way to generate ontology data (e.g., CREAM [7]). Ontology data can also be generated by automatically annotating web pages (e.g. [8]). Although a large amount of ontology data can be generated by this type of methods, some applications may not be able to use them due to its relatively poor quality.

On the other hand, data matching is mostly a research topic in the traditional database community. It involves creating semantic matching between objects, instances, or records from different data sources (mainly different databases) [9]. Different techniques have been employed to perform data matching in different applications (e.g., [10, 11]). Although we can directly use these methods to perform ontology data matching by treating instances in ontology data as data records in databases, this ignores the particular features that ontology data inherently has, which may affect the performance of ontology data matching.

3 Framework Overview

We first give a brief overview of our framework that supports the generation and matching of ontology data. The framework uses the super peer topology [4] for the peer-to-peer architecture. Different from traditional client/server architecture, it shifts part of the tasks of the server (super peer) to the clients (ordinary peers). For example, when processing a query, a super peer can simply tell the query issuer from which peers they can get potential query results instead of returning the complete query results directly.

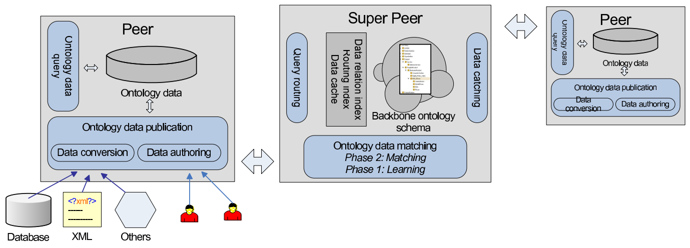


Fig. 1. The framework that supports ontology generation and matching with a design of a peer-to-peer architecture

Accordingly, as shown in Fig. 1, our framework introduces two types of peers: super peer and ordinary peer (or just denoted as peer). The super peer acts as a coordinator for peers connected to it. It hosts the backbone ontology schema used for the generation of ontology data and provides related functions, one of which is *ontology data matching* (Section 5). On the other hand, peers offer functions that support ontology data generation (*ontology data publication*) and query (*ontology data query*).

4 Generation of Ontology Data from Peers

4.1 Data Conversion

As many existing data are stored in databases or formatted in XML with no ontologies to interpret them, it is desirable to convert them into formats that can be explained by given ontology schemas and ready to be integrated. Mostly we use OWL to encode these converted data according to the backbone ontology. Here we only discuss the case of XML data due to limited space.

XQuery [12] is used to convert ordinary XML data into OWL format.Executed by a XQuery-compatible engine, a query written in XQuery takes XML files as input and generates results in an XML format defined by the query itself. Since OWL is also based on XML syntax, the problem is then transformed into how to design the query so that the output results are actually the desired OWL format. The advantage of using XQuery instead of developing our own programs for conversion is obvious. We don't have to design any custom conversion rules and to maintain the program, which might be time-consuming and error-prone. As a W3C candidate recommendation, XQuery is versatile enough to satisfy our needs and has several implemented query engines for us to choose. Therefore, to convert XML data to desired OWL format, we only focus on the design of the queries and use *Nux* (http://dsd.lbl.gov/nux/), a java toolkit capable of XQuery processing, to process them.

4.2 Data Authoring

While existing data is very useful for the ontology data generation, it is equally important to allow new data to be created and published in the framework.

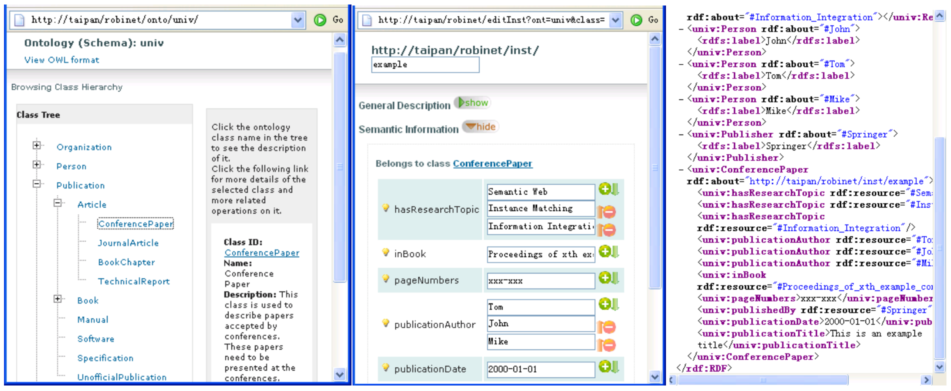
Data authoring is the process during which peers create their own ontology data and publish it into the framework. Like the current Web, where data is directly contributed and published by various individuals and organizations, our framework should also enable different individuals or organizations to publish their new data via their corresponding peers. Therefore, the data in the framework will be very dynamic, often reflecting its very recent status while still retaining reasonable semantics thanks to the backbone ontology schemas.

Users who want to author their concerned data through the peers should be familiar with the backbone ontology schemas. We develop web-based programs at each peer server so that users can get familiar with them easily and quickly. For example, Fig. 2 (a) shows the interface that allows users to learn a backbone ontology schema describing the university settings by browsing intuitively. Therefore, the users don't have to study the original ontology schema encoded in OWL with more efforts. A professor, if he/she wants to publish some data about his/her recent publications, can choose the class that is most appropriate for the data. He/She then can use the selected class to create the ontology data, through a friendly interface as shown in Fig. 2 (b). With the data supplied by the user, the program at the peer server generates the corresponding ontology data in OWL format as shown in Fig. 2 (c). In summary, supported with these functionalities, users can create and publish ontology data with ease.

5 Ontology Data Matching

Data matching process is designed to deal with the problem of implicit relations among the data from different peers. The necessity of it can be illustrated by the following example. Suppose a professor has a peer contributing data about his/her own details including contact information, research interests, supervised students, research groups, selected publications (without details such as abstracts and full texts), and etc. Meanwhile a publisher's peer contributes information of a detailed publication list, which includes some publications (with full texts) of that professor. The publisher's peer lets us know details of publications by that professor. All the data are published as instances of concepts of the given ontology. Because the instance describing the professor from the publisher's peer is not explicitly related to that from the professor's peer due to the decentralized environment, we only get a partial view of the professor's information from these separate peers. It is impossible to issue an enquiry like getting some publication details of a professor whose research interests are of a given area.

Therefore, the task is to match the instances from different peers, making their implicit relations explicit. It is common to compute similarities between instances to determine if they are matched. Several similarity measurements from different aspects are used in the framework. A learning mechanism is employed



```
http://taipan/robinet/onto/univ/

Ontology (Schema): univ
View OWL format

Browsing Class Hierarchy

Class Tree
+ Organization
+ Person
- Publication
    - Article
        ConferencePaper
        JournalArticle
        BookChapter
        TechnicalReport
    + Book
      Manual
      Software
      Specification
      UnofficialPublication

Click the ontology class name in the tree to see the description of it.
Click the following link for more details of the selected class and more related operations on it.

Class ID: ConferencePaper
Name: Conference Paper
Description: This class is used to describe papers accepted by conferences. These papers need to be presented at the conferences.
```

(a)

```
http://taipan/robinet/editInst?ont=univ&class=

http://taipan/robinet/inst/
example

General Description  show
Semantic Information  hide

Belongs to class ConferencePaper
hasResearchTopic     Semantic Web
                     Instance Matching
                     Information Integrati
inBook               Proceedings of xth ex
pageNumbers          xxx-xxx
publicationAuthor    Tom
                     John
                     Mike
publicationDate      2000-01-01
```

(b)

```
  rdf:about="#Information_Integration"></univ:Re
- <univ:Person rdf:about="#John">
    <rdfs:label>John</rdfs:label>
  </univ:Person>
- <univ:Person rdf:about="#Tom">
    <rdfs:label>Tom</rdfs:label>
  </univ:Person>
- <univ:Person rdf:about="#Mike">
    <rdfs:label>Mike</rdfs:label>
  </univ:Person>
- <univ:Publisher rdf:about="#Springer">
    <rdfs:label>Springer</rdfs:label>
  </univ:Publisher>
- <univ:ConferencePaper
  rdf:about="http://taipan/robinet/inst/example">
    <univ:hasResearchTopic rdf:resource="#Sem
    <univ:hasResearchTopic rdf:resource="#Ins
    <univ:hasResearchTopic
    rdf:resource="#Information_Integration"/>
    <univ:publicationAuthor rdf:resource="#To
    <univ:publicationAuthor rdf:resource="#Jo
    <univ:publicationAuthor rdf:resource="#Mi
    <univ:inBook
    rdf:resource="#Proceedings_of_xth_example_co
    <univ:pageNumbers>xxx-xxx</univ:pageNumber
    <univ:publishedBy rdf:resource="#Springer"
    <univ:publicationDate>2000-01-01</univ:pub
    <univ:publicationTitle>This is an example
    title</univ:publicationTitle>
  </univ:ConferencePaper>
</rdf:RDF>
```

(c)

Fig. 2. The user interface for data authoring and the generated data

to implicitly combine these measurements in a meaningful and adaptive way. This involves a learning phase to build the model and an matching phase to apply the model for data matching.

5.1 Learning Phase

The learning phase involves the training of a binary classifier from matched and unmatched instances. Support vector machines [13] are chosen as the classification model in the proposed framework. First, a certain amount of initial data are gathered by super peer from different peers. This data set should contain a portion of matched instances. These matched instances are not discovered and specified initially, while training an SVM classifier requires both specified matched instances and unmatched instances as positive and negative samples. Therefore, these initial data should be checked and tagged manually for the training. An initial similarity checking and sorting process based on selected instances properties is performed to make the manual tagging easier. After all these initial data are tagged and the matched and unmatched instance pairs are created, it is ready to train the classifier.

Several similarity measurements are used to compute similarity/distance scores for instance pairs as different feature scores. string edit distance [14] (denoted as SED) and cosine similarity based on TF-IDF [15] (denoted as SIM) are used to measure the string similarity of instance properties at character level and at word level respectively. In addition to string-based similarities, ontology-based similarities are also used. We define the term of "concept distance" (denoted as CD), which can measure the distance of concepts of two given instances according to the ontology. Instances belonging to the same concepts have the closest distances while those belonging to disjoint concepts have the largest distances. As ontology technology allows instances to be related by object properties [3], it is useful to check the "context similarity" (denoted as CS) for object properties of two instances. If the instances that are related to two target instances via the same object property are similar according to string-based similarities, the two target instances will have high context similarity. Details of these similarity measurements are presented in another paper due to limited space.

Given the above different similarity measurements, we create feature vector for each pair of instances from the initial tagged data set. For a pair of instance a and instance b, its feature vector is composed as follows:

$$\begin{aligned}\mathbf{p}(a,b) = [&SIM_{d_1}(a,b),\ldots,SIM_{d_m}(a,b),\\ &SED_{d_1}(a,b),\ldots,SED_{d_m}(a,b),\\ &CS_{o_1}(a,b),\ldots,CS_{o_n}(a,b),\\ &CD(a,b)].\end{aligned} \tag{1}$$

where $d_1,\cdots,d_m$ are data type properties [3] of the instances; and $o_1,\cdots,o_n$ are object properties. These feature vectors together with the tagged information (matched or unmatched) enable us to build the classification model for the matching.

5.2 Matching Phase

During the Matching phase, the super peer matches instances from the peers when they contribute their own data. When a peer has some data published, the super peer will be notified. Instances in those data will be sent to the super peer for an initial check upon its request. The initial check searches potentially matched instances that are previously indexed for the new instances through an inverted index. If no instances are found for the new instances or the found instances have very low hits, these new instances will be ignored. This initial check screens off a number of instances. For those instances with potentially matched instances found, instance pairs are created as the input of the classifier.

The trained SVM classifier is used to determine if the instance pairs are matched pairs with the following classification function:

$$f(\mathbf{q}) = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{p_i},\mathbf{q}) + b \tag{2}$$

where $K(\mathbf{p},\mathbf{q})$ is a kernel function used for mapping features into different spaces, α_i is the Lagrangian coefficient of the i-th training instance pair, $y_i \in \{-1,+1\}$ is the label of the training instance pair. In this function, α_i, b are obtained during the learning phase and $f(\mathbf{q})$ indicates the distance of $\mathbf{q}$ from the optimal hyperplane. So we can use this value to evaluate the confidence level of the pair being matched [11]. That is, if $f(\mathbf{r}) > f(\mathbf{q})$, then $\mathbf{r}$ is more likely to be a matched pair than $\mathbf{q}$. For a potentially matched instance pair $\mathbf{q}$, we regard $\mathbf{q}$ as a matched pair if $f(\mathbf{q}) > \delta$, where δ is obtained from experiments. This δ allows the classifier to achieve the maximum F measure [16] in the cross validation.

After matched instance pairs are found through the above process, an index storing data relation information among peers is updated by adding information about these pairs. This index reveals the semantic relations of the data across the peers. It is therefore possible to query related information from more than one peer.

6 Experiments

Experiments are conducted to test the effectiveness of ontology data matching. Data related to the university setting (Professors, Publications, and etc) are collected from five different sources over the Web. As these existing data are in various formats, data conversion has been performed to make them aligned to a backbone ontology schema that describes the university setting. Totally there are 453 instances in the data set. After manually checking these instances, 136 instances are found to match with each other. Instance pairs including matched and unmatched pairs are then generated from these tagged instances. The SVM^{light} package [17] is used in our experiments. 20 random experimental trials are conducted. For one trial we split the pair set into two folds randomly, one for training, the other for testing and then reverse. Traditional measurements such as *precision*, *recall* and *F measure* [16] are used for evaluation. We record the maximum F measure achieved in each trial and its corresponding recall and precision. The overall results, shown in table 1, are obtained by averaging all these trials. The first row indicates the different methods of similarity measurements (or their combinations) used in creating the feature vectors. "ONTO" indicates the features related to ontology (CD, CS) are used. Overall, the method that explores the ontology features yields the best result.

Table 1. Overall results of different methods of similarity measurements used for ontology data matching

Similarity	SIM	SED	SIM+SED	SIM+SED+ONTO
Precision	0.909	0.945	0.919	**0.929**
Recall	0.910	0.752	0.933	**0.946**
F measure	0.909	0.837	0.926	**0.937**

7 Conclusions and Future Work

This paper proposes a framework that supports the generation and matching of ontology data in a peer-to-peer environment. It helps users generate ontology data in two ways. Besides data generation, the issue of ontology data matching in the peer-to-peer environment is also addressed . Experiments show that the proposed matching method which explores the ontology features outperforms other traditional methods. With a matching process that employs this method in the framework, ontology data across peers can be interrelated to offer better information services.

Future work includes the refinement of the data matching process and the design of particular query services based on interrelated ontology data after matching. Since the ontology data matching method can not completely guarantee correct decisions, it is desirable to incorporate peer interaction to correct them. Given the interrelated data across different peers, particular query or reasoning services will be designed to take advantage of them.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5) (2001) 34–43
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The description logic handbook : theory, implementation, and applications. Cambridge University Press, New York (2002)
3. McGuinness, D.L., Harmelen, F.v.: Owl web ontology language overview. w3c recommendation. http://www.w3.org/TR/owl-features/ (2004)
4. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmer, M., Risch, T.: Edutella: a p2p networking infrastructure based on rdf. In: WWW 2002. ACM Press, Honolulu, Hawaii, USA (2002) 604–615
5. Halevy, A.Y., Ives, Z.G., Mork, P., Tatarinov, I.: Piazza: data management infrastructure for semantic web applications. In: WWW2003. (2003) 556–567
6. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca-Grau, B., Hendler, J.: Swoop: A 'web' ontology editing browser. Journal of Web Semantics **4**(2) (2006) 144–153
7. Handschuh, S., Staab, S., Maedche, A.: Cream: creating relational metadata with a component-based, ontology-driven annotation framework. In: Proceedings of the international conference on Knowledge capture. ACM Press (2001) 76–83
8. Cimiano, P., Handschuh, S., Staab, S.: Towards the self-annotating web. In: Proceedings of the 13th international conference on World Wide Web. (2004) 462–471
9. Doan, A., Halevy, A.Y.: Semantic-integration research in the database community. AI Mag. **26**(1) (2005) 83–94
10. Tejada, S., Knoblock, C.A., Minton, S.: Learning domain-independent string transformation weights for high accuracy object identification. In: KDD '02, New York, NY, USA, ACM Press (2002) 350–359
11. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: KDD '03, New York, NY, USA, ACM Press (2003) 39–48
12. Boag, S., Chamberlin, D., Fernndez, M.F., Florescu, D., Robie, J., Simeon, J.: Xquery 1.0: An xml query language. http://www.w3.org/TR/xquery (2006)
13. Vapnik, V.N.: The nature of statistical learning theory. 2nd edn. Statistics for engineering and information science. Springer, New York (1999)
14. Gusfield, D.: Algorithms on strings, trees, and sequences : computer science and computational biology. Cambridge University Press (1997)
15. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. **24**(5) (1988) 513–523
16. Baeza-Yates, R., Ribeiro, B.d.A.N.: Modern information retrieval. Addison-Wesley Longman, Reading, Mass. (1999)
17. Joachims, T.: Text categorization with suport vector machines: Learning with many relevant features. In: ECML '98: Proceedings of the 10th European Conference on Machine Learning, London, UK, Springer-Verlag (1998) 137–142