Integration of Ontology Data through Learning Instance Matching

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

Information integration with the aid of ontology can roughly be divided into two levels: schema level and data level. Most research has been focused on the schema level, i.e., mapping/matching concepts and properties in different ontologies with each other. However, the data level integration is equally important, especially in the decentralized Semantic Web environment. Noticing that ontology data (in the form of instances of concepts) from different sources often have different perspectives and may overlap with each other, we develop a matching method that utilizes the features of ontology and employs the machine learning approach to integrate those instances. By exploring ontology features, this method performs better than other general methods, which is revealed in our experiments. Through the process that implements the matching method, ontology data can be integrated together to offer more sophisticated services.

1 Introduction

The Semantic Web aims at creating a platform where information has its semantics and can be understood and processed by computers themselves with minimum human interference. Ontology theory and its related technology have been developed to help construct such a platform because ontology promises to encode certain levels of semantics for information and offers a set of common vocabulary for people or computer to communicate with. An ontology can roughly be divided into two parts. One part is schema including concepts, properties, relations and etc and another part is data consisting of instances of concepts, or individuals. Ontology matching, mostly dealing with matching concepts that come from different but similar ontologies with each other [7], is important for data integration, particularly data transformation. however, this schema-level integration can not guarantee that data itself transformed from one source be semantically linked to data from another source. Data-level integration is then necessary. The need for data-level integration can be explained as follows. Since the Semantic Web is a distributed environment, we picture it as an environment consisting of numerous information sources or peers. For a particular domain, there could be one or more ontologies that encode the knowledge of this domain in the form of concepts, properties and their semantic relations. Let's assume only one such backbone ontology for one domain. This ontology is actually recognized as Terminology Box (TBox) in description logics [1]. For such a domain ontology, a single data source can hardly offer entire or enough data in the form of individuals or instances for the concepts of the ontology. In reality, each data source provides a portion of data that is aligned to a certain facet of the ontology. Overlapping can happen when several data sources describe the same instances (likely from different angels). While it is easy to maintain relations of instances within each data source, it is difficult to create/maintain these relations between them due to the decentralization and autonomy of data sources.

We deal with this problem using ontology related techniques in conjunction with machine learning methods for instance matching. As assumed, in our method one backbone domain ontology is used to align data. The core task is to integrate those data segments from different sources by matching the instances contained in those segments. This matching process is not trivial because information of instances by different sources may not be exact for the same real world entities, and even conflictions may occur. Additionally, instances that seems to be the same sometimes may actually refer to different entities. So we not only use traditional string based similarity measurements but also create novel ontology enhanced measurements to check instance similarity. A support vector machine (SVM) [13] classifier, trained with these similarity measures, is used to identify instances referring to same real entities, which enables us to create semantic relations between different data sources. Experiments reveal that the use of ontology features increases accuracy of instance matching for data integration.

The rest of the paper is organized as follows. Section

2 discusses related works in the field of ontology-based information integration. A brief description of ontology, its schema and data are given in Section 3 . Section 4 presents the similarity measurements used in instance matching, and Section 5 discusses the machine learning method for instance matching by using the SVM classifier. Section 6 concludes the paper and discusses the further study.

2 Related Works

Research on ontology-based information integration starts with the representation of ontology itself (Description logic [1], OWL [10] and etc.). Several frameworks/systems for ontology-based information integration have been proposed (refer to [14] for a review in this area). However, their research efforts are mainly focused on the integration at the ontology schema level. For those who consider integrating data at the ontology data level by creating/maintain data relations, they provide general models. No specific or automatic methods are discussed in details. In addition, to our knowledge, the issue of decentralization that causes the difficulty of data-level integration hasn't been thoroughly studied and no specific methods are proposed so far.

On the other hand, related studies on data matching, object identification or record linkage haven been conducted for a while in database research community. These studies try to decide if two relational tuples from two sources refer to the same real world entity [5]. Several techniques have been employed to perform data matching in different applications. String edit distance [8] and TF-IDF based methods [11] are two common measurements to calculate the similarity of data records for matching. To make the measurements adaptive for dynamic situations, machine learning techniques have been incorporated. Bilenko and Mooney [3] use a stochastic model and SVM classifier to learn string similarity measures from samples so that accuracy can be improved for the given situation. Tejada et al. [12] use a mapping-rule learner consisting of a committee of decision tree classifiers and a transformation weight learner to help create mapping between objects from different sources. McCallum et al [9] employ clustering-based methods to identify duplicated reference records. Besides calculating or learning similarity among shared attributes of objects/records, profile-based object matching method proposed by Doan et al. [6] also correlates disjoint attributes to improve matching accuracy. However, the features of ontology haven't been fully explored to develop effective methods for ontology data matching, which is one of the key issues of ontology-based data-level integration.

3 Ontology: Schema and Data

Ontology can be expressed in description logics (DL) [1], which are a well-known family of knowledge representation mechanism and have been studied for several years. An ontology can be regarded as a typical DL knowledge base that consists of two components: a "TBox" and a "ABox". Concepts and their relations of the ontology are defined in the TBox as a set of asserted axioms. Individuals (i.e., instances of concepts) are contained in the ABox.

To make ontologies easy to be published on the Web, World Wide Web Consortium has proposed a DL-based web ontology language: OWL [10]. It provides a set of vocabulary as constructs, enabling people to define concepts, properties, individuals, and their relations. Typically, property in OWL has two categories: data type property and object property. Data type properties allow people to describe specific attributes of a concept, such as "age of person", "address of company". Meanwhile, object properties enable people to link two concepts with a semantic relation, like "supervise" between "professor" and "student".

Corresponding to the notions of TBox and ABox in DL, ontology encoded in OWL can also be partitioned into two parts: ontology schema and ontology data. Definitions of concepts, properties and their relations in the owl file(s) are treated as ontology schema. Instances of these concepts, or individuals, are treated as ontology data.

4 Instance Matching

Instance matching tries to find out instances from different data sources referring to the same real world entities, enabling people to create links between the sources for the purpose of integration. It can be performed by checking similarities between instances. If the similarity degree of two instances reaches a certain level, then the two instances can be regarded as matched. There are two major methods to compute similarity degrees between instances: string edit distance [8] and cosine similarity based on TF-IDF [11]. Besides the two methods, we develop an extended method for instance matching, in which ontology features are explored.

4.1 String Edit Distance and TF-IDF

String edit distance (also known as Levenshtein distance) is used to compare string similarity at the character level. It is defined as the minimum cost of transforming one string into another by insertions, deletions, or substitutions. For two strings S_1 and S_2 with length m and n respectively, their string edit distance $SED(S_1, S_2)$ can be computed by the dynamic programming algorithm [8].

Unlike string edit distance methods that compute the distances of string at character level, TF-IDF based methods use a vector space model [2], treating strings as "bag of tokens" and ignoring the sequential order of tokens in the strings. An instance that consists of one or more strings can be viewed as a virtual document containing a bag of string tokens. If there are N instances, then the corresponding N virtual documents form a virtual corpus, which may finally have a vocabulary of n distinct tokens. A sparse n-dimensional token vector V_i can be derived from the i-th virtual document with each element $v_{i,j}$ having TF-IDF value computed as follows:

$$v_{i,j} = TF_{i,j} \times \log \frac{N}{DF_j} \tag{1}$$

where $TF_{i,j}$ is the frequency of token t_j in the i-th document and DF_j is the frequency of the document that contains the token t_j.

For two instances with S and T as their derived token vectors respectively, the similarity between them is computed as normalized dot product between their corresponding token vectors.

$$SIM(S,T) = \frac{\sum_{j=1}^{n} s_j \cdot t_j}{\|S\| \cdot \|T\|}. \tag{2}$$

4.2 Extended Matching by Exploring Ontology Features

The use of ontology enables us to enhance the methods that only compute string similarities between ontology instances. Two ontology features are used in the proposed extended method.

The first feature is the ontology hierarchy. Given an ontology schema, we can compute the subsumption relations between concepts in the ontology schema using a specific reasoner. Then hierarchy of the concepts can be constructed, which allows us to explore the "concept-level similarity" of instances. Because ontology data are contributed by different data sources separately, the quality and the focus of completeness of the data vary. It can not be guaranteed that the instances referring to the same real world entity are identified exactly with the same concept by different data sources. However, their concepts should have some relations according to their common ontology schema. For example, if two instances from different data sources are identified as instances of concept "`Student`" and "`GradateStudent`" respectively, then they are more likely to be the same than two instances with one identified as "`Student`" and another as "`Professor`", given the two pairs have the same similarity degree measured by string edit distance or TF-IDF. In addition, we are able to define disjoint concepts in the ontology schema. If "`Student`" and "`Professor`" (both are sub concept of "`Person`") are defined as disjoint concepts, and we assume that each publisher of the data source is aware of this assertion when contributing ontology data, then a student instance could never be matched with a professor instance, even they have very high string-based similarity.

We define "concept distance" to measure concept-level similarity. Suppose two instances i, j are of concept A and B respectively. This can be denoted as $A(i)$, $B(j)$. Concept distance between i and j, denoted by $CD(i,j)$, is defined as follows:

$$CD(i,j) = \begin{cases} 0 & A \equiv B, \\ PT(A,B) & A \sqsubseteq B \text{ or } B \sqsubseteq A, \\ PT(A,B) + P & A \not\sqsubseteq B, B \not\sqsubseteq A, \\ \infty & A \sqcap B = \perp . \end{cases} \tag{3}$$

where $PT(A,B)$ means the length of the path between concept A and B according to the computed concept hierarchical tree; P is a penalty item, always given a positive number. In our experiments, $P = 2$. For ∞, we use a big enough positive number to represent it in the implementation. Intuitively, if the concept distance of two instances are bigger, the likeness of being same would be less.

We also examine object properties of instances to compute "context similarity". Object properties allow users to create specific relations between instances. Usually, before an object property is employed to link instances with semantic relations, it is defined between concepts with an option to specify its cardinality constraints. In addition, an object property can have an inverse object property, which allows more flexible ways of describing ontology data. Using inverse object properties could be very common among different data sources. For example, a publisher may tend to describe publications using a property "`wrttenBy`" to relate them to their author instances, while a professor may choose to use the inverse property "`write`" to link his own instance with his publications.

By reasoning on the inverse properties and checking the cardinalities on them, we can compute the context similarity between instances. For instance a and b, their context similarity in terms of the object property r, denoted by $CS_r(a,b)$, is computed as follows:

$$CS_r(a,b) = \frac{\sum_{\forall m, r(a,m)} \sum_{\forall n, r(b,n)} SIM(m,n)}{|\{m : r(a,m)\}| \times |\{n : r(b,n)\}|} \tag{4}$$

5 Learning Instance Matching

We employ a machine learning approach to integrate these different similarity measurements for instance matching. We create a set of matched instance pairs with positive labels and a set of non-matched instance pairs with negative labels. A binary classifier is trained by using different similarity measurements as features from the two pair sets. This binary classifier then acts as a paring function [4] $h(a, b)$, taking a pair of instances a, b as input and generating decision values as output. If it generates positive values, the two input instances are regarded as matched; otherwise, unmatched.

Based on the discussion in [3], we choose support vector machine (SVM) [13] as the classification model. SVM is able to learn from small training sets of high-dimensional data with satisfactory precision. Therefore, we create a feature vector of an instance pair using the separate property similarity measures for SED and SIM instead of their overall similarity measures. For a pair of instance a and instance b, its feature vector is composed as follows:

$$\begin{aligned}\mathbf{p}(a,b) \quad = \quad & [SIM_{d_1}(a,b), \ldots, SIM_{d_m}(a,b), \\ & SED_{d_1}(a,b), \ldots, SED_{d_m}(a,b), \\ & CS_{o_1}(a,b), \ldots, CS_{o_n}(a,b), \\ & CD(a,b)]. \qquad (5)\end{aligned}$$

Although using separate property features increases the dimensions of the feature vectors greatly, the learned classifier can implicitly reflect the weight of different properties based on the training set, relying on more informative properties to make classification decisions.

Experiments have been conducted to test the proposed method for learning instance matching. 453 real word instances (aligned with a backbone ontology schema describing the university domain) were collected from different data sources. The results in the form of precision and recall (illustrated in Table 1) show that performance of instance matching was improved by incorporating ontology features (due to limited space, details of experiments are omitted in this paper).

Table 1. Precision/recall of different methods

Recall Level		0.2	0.4	0.6	0.8
	SIM	0.883	0.905	0.930	0.927
Precision	SED	0.939	0.966	0.970	0.453
	+ONTO	**0.959**	**0.978**	**0.984**	**0.968**

6 Conclusions and Further study

Ontology-based information integration involves two levels: schema level and data level. The data level integration is very necessary in the real world applications. This study proposes a method to integrate ontology data from different sources by learning to match instances of concepts contained in those data. Experiments confirm that by incorporating ontology features, performance of instance matching can be improved.

By the instance matching process, ontology data can be integrated together to offer more sophisticated services. Therefore, the future work of this study includes the design of a system that uses this matching process to integrate ontology data and the design of query or reasoning services that take advantage of the integrated data. It is believed that services provided by this system will be more powerful since it can access multiple sources via the semantic relations created by instance matching.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, New York, 2002.

[2] R. Baeza-Yates and B. d. A. N. Ribeiro. *Modern information retrieval*. Addison-Wesley Longman, 1999.

[3] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD '03*, pages 39–48, New York, NY, USA, 2003. ACM Press.

[4] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *KDD '02*, pages 475–480, 2002.

[5] A. Doan and A. Y. Halevy. Semantic-integration research in the database community. *AI Mag.*, 26(1):83–94, 2005.

[6] A. Doan, Y. Lu, Y. Lee, and J. Han. Profile-based object matching for information integration. *IEEE Intelligent Systems*, 18(5):54–59, 2003.

[7] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *WWW '02*, pages 662–673. 2002.

[8] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, 1997.

[9] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD '00*, pages 169–178, New York, NY, USA, 2000. ACM Press.

[10] D. L. McGuinness and F. v. Harmelen. Owl web ontology language overview. w3c recommendation. http://www.w3.org/tr/owl-features/, 2004.

[11] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[12] S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *KDD '02*, pages 350–359, New York, NY, USA, 2002. ACM Press.

[13] V. N. Vapnik. *The nature of statistical learning theory*. Statistics for engineering and information science. Springer, New York, 2nd edition, 1999.

[14] H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner. Ontology-based integration of information-a survey of existing approaches. In *IJCAI-01 Workshop: Ontologies and Information Sharing*, pages 108–117. Seattle, WA, 2001.