Knowledge-Based Systems 50 (2013) 112–120

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Large scale instance matching via multiple indexes and candidate selection

Juanzi Li [a], Zhichun Wang [a,b,*], Xiao Zhang [a], Jie Tang [a]

[a] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[b] College of Information Science and Technology, Beijing Normal University, Beijing, China

ARTICLE INFO

Article history:
Received 16 November 2011
Received in revised form 30 May 2013
Accepted 6 June 2013
Available online 18 June 2013

Keywords:
Heterogeneous data
Semantic web
Instance matching
Ontology matching
Linked data

ABSTRACT

Instance matching aims to discover the linkage between different descriptions of real objects across heterogeneous data sources. With the rapid development of Semantic Web, especially of the linked data, automatically instance matching has been become the fundamental issue for ontological data sharing and integration. Instances in the ontologies are often in large scale, which contains millions of, or even hundreds of millions objects. Directly applying previous schema level ontology matching methods is infeasible. In this paper, we systematically investigate the characteristics of instance matching, and then propose a scalable and efficient instance matching approach named VMI. VMI generates multiple vectors for different kinds of intained in the ontology instances, and uses a set of inverted indexes based rules to get the primary matching candidates. Then it employs user customized property values to further eliminate the incorrect matchings. Finally the similarities of matching candidates are computed as the integrated vector distances and the matching results are extracted. Experiments on instance track from OAEI 2009 and OAEI 2010 show that the proposed method achieves better effectiveness and efficiency (a speedup of more than 100 times and a bit better performance (+3.0% to 5.0% in terms of F1-score) than top performer RiMOM on most of the datasets). Experiments on Linked MDB and DBpedia show that VMI can obtain comparable results with the SILK system (about 26,000 results with good quality).

1. Introduction

Ontology is one of the key components to realize the Semantic Web. With the rapid development of the Social Web, a lot of ontologies especially lightweight ontologies have been widely used, and a huge number of instances were annotated according to the ontologies. For example, the FOAF vocabulary (schema) is comprised of 13 classes and 60 properties while LiveJournal website alone provides approximately 15,000,000 FOAF profiles (instances). The DBpedia ontology, which covers 273 classes described by 1300 different properties, contains about 1,600,000 instances. GeoNames provides RDF descriptions of more than 6,500,000 geographical features worldwide. The Linking Open Data (LOD) project already has a dataset of more than 4.7 billion RDF triples and around 142 million RDF links between instances [6].

As the number of published ontologies grows, a increasing number of ontology-based applications have also been proposed, such as Question Answering [14], Query Expansion [28], Knowledge Support [1] and Web Services [26]. The rapid usage of ontologies arises the ontology heterogeneity problem. In the last decade, ontology matching has been widely studied as the key technology to reach interoperability over ontologies [20,8,24]. Traditionally, ontology matching approaches focus on finding semantic correspondences between complex ontology schemas. Recently, as the number of ontology instances grows rapidly, the problem of instance matching attracts increasingly more research interest [35]. The yearly ontology matching competition OAEI (Ontology Alignment Evaluation Initiative)[1] has set up instance matching campaigns since 2009. Several systems, such as RiMOM [36], HMatch [7], and FBEM [30], have participated in the instance matching tasks of OAEI. The problem of instance matching involves handling large number of instances, which raise new challenges: (1) How to deal with large scale input? Shvaiko and Enzuenat [29] point out that the scalability is important for ontology matching approaches. Widely used matching techniques such as Edit Distance [15], KNN [16], Google Distance [12] will take much running time when applied to large number of instances. Suppose we apply the Edit Distance, one of the most efficient similarity metrics, to match two ontologies with 1,000,000 instances. Even on a server with 32 Gigabyte memory and 3.2 GHz CPU, the running time of

* Corresponding author at: Department of Computer Science and Technology, Tsinghua University, Beijing, China. Tel.: +86 01062773618; fax: +86 010 62781461.
E-mail addresses: ljz@keg.cs.tsinghua.edu.cn (J. Li), zcwang@bnu.edu.cn (Z. Wang), zhangxiao@keg.cs.tsinghua.edu (X. Zhang), jietang@tsinghua.edu.cn (J. Tang).

[1] http://oaei.ontologymatching.org/.

0950-7051/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2013.06.004

calculating all the potential matches will be up to 2 days. (2) How to trade off between precision and recall? Most approaches use strict measures to find matching results with very high precision but only a very small part of the potential matches are obtained. By investigating information contained in ontology instances, we observe that there are several different characteristics of instance matching compared with the traditional ontology matching in schema level. Firstly, instance data is usually with large scale. Secondly, instance data contains devious semantic information. Usually concepts and properties in ontology schema are described with labels and comments. However, for instances or individuals in an ontology, every property is given a specific value and represented in various ways. For example, the e-mail address of a person, the ISBN number of a book, the DNA sequence of a gene is consisted of a large number of different values of validated types. It is difficult to take full advantages of the information. Thirdly, the concepts and properties in the ontology schema construct a connected graph structure. The ontology can be viewed as a whole ontology graph and some graph-based algorithms are employed in ontology matching. However, a concept may have lots of instances and all the instances are with almost the same structure. The graph algorithm with the whole ontology graph is not suitable for the instance matching task.

To address the above two challenges, we propose a large scale instance matching method (named VMI) by using multiple indexes and candidate selection. VMI aims at matching large scale instance datasets efficiently and generates as many matching results as possible with high quality. In particular, VMI uses the vector space model to represent instances' descriptive information. VMI creates two types of vectors for each instance, one for names and labels of the instance and the other for descriptive information and information from neighboring instances. We build inverted indexes for these types of vectors and select matching candidates according to the indexes. In this way, VMI is able to avoid pair-wise comparison and reduces the matching space greatly so that the matching efficiency can be improved. Then VMI compares the value pairs from user specified properties to filter the primary candidates and improve the precision. Experimental results on the datasets from the Instance Matching track of OAEI 2009 and OAEI 2010 show that VMI is much faster than existing methods (100 times faster than the participants) while achieves better (+3.0% to 5.0% in terms of F1-score on most of datasets) accuracy performance than the top performer RiMOM. Experimental results on LinkedMDB and DBpedia dataset show that VMI can generate matching results with almost the same amount and quality as ones from the SILK system [33].

Contributions of this work can be summarized as:

- We formally define the problem of large scale instance matching.
- We propose an efficient and accurate instance matching method by using the inverted indexing and candidate matching selection rules.
- We validate the proposed VMI approach on three datasets from the Instance Matching track of OAEI 2009 and 2010 as well as datasets from Linked Open Data. Experimental results show that VMI can achieve a more than 100 $\times$ speedup than the best performer in OAEI 2009 and a comparable precision and recall performance with the prevalent SILK system.

The rest of this paper is organized as follows. In Section 2, some related work are summarized. In Section 3, we give some preliminary and definitions. In Section 4, we show an overview of VMI algorithm and a detailed presentation of VMI is given in Section 5. Experimental results are illustrated in Section 6 with discussions. Finally, conclusion and future work are given in Section 7.

2. Related work

There has been already several approaches dealing with the instance matching problem. Most of them focus on achieving high precision and recall, the problem of matching large scale instances has not been well studied. We summarize some of these methods and systems compare them with our proposed approach VMI.

COMA++ [3] is an schema and ontology matching tool utilizing a composite approach to combine different match algorithms. In the enhanced eversion of COMA++, it uses two methods to matching instances [10]: one is the content-based similarity, the other is constraint-based similarity. COMA++ needs to compare all the instances between two ontologies, and it also uses a similarity propagation algorithm to transfer similarities from instances to their surrounding ontology elements. Our approach generates a virtual document for each instance to include its neighboring information, and computes the similarity by using Vector Space Model; it is more efficient than the iterative similarity propagation. Furthermore, our approach selects the matching candidates based on two inverted indexes, it does not need to compare all the instance pairs. HMatch [7] is a ontology matching suite which provides a component for instance matching. In HMatch, each instance is represented as a tree where role fillers are nodes and roles are labeled edges. Matching is performed by traversing the instances trees of the two instance in postorder, and recursively executing filler similarity. Filler similarities of different properties are combined by weighted averaging with manually weights. RiMOM [31]21 uses a systematic approach to quantitatively estimate the similarity characteristics for each matching task and employs a strategy selection method to automatically combine the matching strategies based on two estimated factors. For instance matching, RiMOM chooses some data-type properties as the “necessary” and “sufficient” attributes manually. “sufficient” Attributes are use to find the initial alignment while the “necessary” attributes for refinement, and a similarity propagation is employed in the last step. DSSim [23] is an ontology mapping system used with a multi-agent ontology mapping framework in the context of question answering. In order to improve the matching quality, it incorporates the Dempster Shafer theory of evidence into the mapping process. DSSim assesses similarity of all the entities from two different ontologies belief combination process of DSSim is also computationally expensive. Therefore, DSSim employs an multi-agent architecture to enable distributed execution of the approach. Our approach uses inverted indexes to select matching candidates, therefore reduces the computation time; it is different from the distributed execution of DSSim that needs multiple machines. FBEM [30] is a feature based instance matching system. It does not need any kind of schema or strong typing information of the instances. FBEM supports a complete generic way to match instances. Given two instances, FBEM first computes the Levenstein similarity between all the features of them, and then calculates the combined similarity score by summing all the maximum similarity feature combinations between two instances. FBEM also implemented a “brute-force” matching, similarity of any instance pairs need to be computed to get the matching results. Being different from FBEM, our approach allows users to specify instance types and important properties to improve the accuracy and efficiency of VMI. SILK [33] is a link discovery engine which automatically finds RDF links between datasets. Users should specify which type of RDF links should be discovered between the data sources as well as which conditions data items must fulfill in order to be interlinked. These link conditions can apply different similarity metrics to multiple