Learning to match ontologies on the Semantic Web

AnHai Doan¹, Jayant Madhavan², Robin Dhamankar¹, Pedro Domingos², Alon Halevy²

¹ Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
e-mail: {anhai,dhamanka}@cs.uiuc.edu
² Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195, USA
e-mail: {jayant,pedrod,alon}@cs.washington.edu

Edited by B.V. Atluri, A. Joshi, and Y. Yesha. Received: December 16, 2002 / Accepted: April 16, 2003
Published online: September 17, 2003 – © Springer-Verlag 2003

Abstract. On the Semantic Web, data will inevitably come from many different ontologies, and information processing across ontologies is not possible without knowing the semantic mappings between them. Manually finding such mappings is tedious, error-prone, and clearly not possible on the Web scale. Hence the development of tools to assist in the ontology mapping process is crucial to the success of the Semantic Web. We describe GLUE, a system that employs machine learning techniques to find such mappings. Given two ontologies, for each concept in one ontology GLUE finds the most similar concept in the other ontology. We give well-founded probabilistic definitions to several practical similarity measures and show that GLUE can work with all of them. Another key feature of GLUE is that it uses multiple learning strategies, each of which exploits well a different type of information either in the data instances or in the taxonomic structure of the ontologies. To further improve matching accuracy, we extend GLUE to incorporate commonsense knowledge and domain constraints into the matching process. Our approach is thus distinguished in that it works with a variety of well-defined similarity notions and that it efficiently incorporates multiple types of knowledge. We describe a set of experiments on several real-world domains and show that GLUE proposes highly accurate semantic mappings. Finally, we extend GLUE to find complex mappings between ontologies and describe experiments that show the promise of the approach.

Keywords: Semantic Web – Ontology matching – Machine learning – Relaxation labeling

1 Introduction

The current World Wide Web has well over 1.5 billion pages [19], but the vast majority of them are in human-readable format only (e.g., HTML). As a consequence, software agents (softbots) cannot understand and process this information, and much of the potential of the Web has so far remained untapped.

In response, researchers have created the vision of the Semantic Web [3], where data has structure and ontologies describe the semantics of the data. When data are marked up using ontologies, softbots can better understand the semantics and therefore more intelligently locate and integrate data for a wide variety of tasks. The following example illustrates the vision of the Semantic Web.

Example 1.1. Suppose you want to find out more about someone you met at a conference. You know that his last name is Cook and that he teaches Computer Science at a nearby university, but you do not know which one. You also know that he just moved to the US from Australia, where he had been an associate professor at his alma mater.

On the World Wide Web of today, you would have trouble finding this person. The above information is not contained within a single Web page, thus making keyword search ineffective. On the Semantic Web, however, you should be able to quickly find the answers. A marked-up directory service makes it easy for your personal softbot to find nearby computer science departments. These departments have marked up data using some ontology such as the one in Fig. 1a. Here the data is organized into a taxonomy that includes courses, people, and professors. Professors have attributes such as name, degree, and degree-granting institution (i.e., the one from which a professor obtained his or her Ph.D. degree). Such marked-up data make it easy for your softbot to find a professor with the last name Cook. Then by examining the attribute “granting institution”, the softbot quickly finds the alma mater CS department in Australia. Here the softbot learns that the data have been marked up using an ontology specific to Australian universities, such as the one in Fig. 1b, and that there are many entities named Cook. However, knowing that “associate professor” is equivalent to “senior lecturer”, the bot can select the right subtree in the departmental taxonomy and zoom in on the old homepage of your conference acquaintance.

The Semantic Web thus offers a compelling vision, but it also raises many difficult challenges. Researchers have been actively working on these challenges, focusing on fleshing out the basic architecture, developing expressive and efficient ontology languages, building techniques for efficient marking up of data, and learning ontologies (e.g., [20,5,36,28,22]).

A key challenge in building the Semantic Web, one that has received relatively little attention, is finding semantic mappings among ontologies. Given the decentralized nature of the
development of the Semantic Web, there will be an explosion in the number of ontologies. Many of these ontologies will describe similar domains, but using different terminologies, and others will have overlapping domains. To integrate data from disparate ontologies, we must know the semantic correspondences among their elements [3, 46]. For example, in the conference-acquaintance scenario described earlier, in order to find the right person, your softbot must know that "associate professor" in the US corresponds to "senior lecturer" in Australia. Thus, the semantic correspondences are in effect the "glue" that holds the ontologies together into a "web of semantics". Without them, the Semantic Web is akin to an electronic "glue" that holds the ontologies together into a "web of semantics".

2 Overview of our solution

In response to the challenge of ontology matching on the Semantic Web, we have developed the GLUE system, which applies machine learning techniques to semiautomatically create semantic mappings. Since taxonomies are central components of ontologies, we focus first on finding one-to-one (1-1) correspondences between the taxonomies of two given ontologies: for each concept node in one taxonomy, find the most similar concept node in the other taxonomy.

Similarity definition: The first issue we address is the meaning of similarity between two concepts. Clearly, many different definitions of similarity are possible, each being appropriate for certain situations. Our approach is based on the observation that many practical measures of similarity can be defined based solely on the joint probability distribution of the concepts involved. Hence, instead of committing to a particular definition of similarity, GLUE calculates the joint distribution of the concepts and lets the application use the joint distribution to compute any suitable similarity measure.

Specifically, for any two concepts A and B, the joint distribution consists of \( P(A, B) \), \( P(A, \overline{B}) \), \( P(\overline{A}, B) \), and \( P(\overline{A}, \overline{B}) \), where a term such as \( P(A, B) \) is the probability that an instance in the domain belongs to concept A but not to concept B. An application can then define similarity to be a suitable function of these four values. For example, a similarity measure we use in this paper is \( P(A \cap B) / (P(A \cup B) \), otherwise known as the Jaccard coefficient [47].

Computing similarities: The second challenge we address is that of computing the joint distribution of any two given concepts A and B. Under certain general assumptions (discussed in Sect. 5), a term such as \( P(A, B) \) can be approximated as the fraction of data instances (in the data associated with the taxonomies or, more generally, in the probability distribution that generated the data) that belong to both A and B. Hence the problem reduces to deciding for each data instance if it belongs to \( A \cap B \). However, the input to our problem includes instances of A and instances of B in isolation. GLUE addresses this problem using machine learning techniques as follows: it uses the instances of A to learn a classifier for A and then classifies instances of B according to that classifier, and vice versa. Thus we have a method for identifying instances of \( A \cap B \).

Multistrategy learning: Applying machine learning to our context raises the question of which learning algorithm to use and which types of information to exploit. Many different types of information can contribute to the classification of an instance: its name, value format, and the word frequencies in its value, and each of these is best utilized by a different learning algorithm. GLUE uses a multistrategy learning approach [12]: we employ a set of learners and then combine their predictions using a metalearner. In previous work [12], we have shown that multistrategy learning is effective in the context of mapping between database schemas.

Exploiting domain constraints: GLUE also attempts to exploit available domain constraints and general heuristics to improve matching accuracy. An example heuristic is the observation that two nodes are likely to match if nodes in their neighborhood also match. An example of a domain constraint is "if node X matches Professor and node Y is an ancestor of X in the taxonomy, then it is unlikely that Y matches Assistant-Professor". Such constraints occur frequently in