Ontology Matching and Schema Integration using Node Ranking

Asankhaya Sharma
Department of Computer Science and Engineering
National Institute of Technology Warangal
Warangal, AP, India

Dr. D.V.L.N. Somayajulu
Department of Computer Science and Engineering
National Institute of Technology Warangal
Warangal, AP, India

Abstract - *In this paper, we present a new way towards ontology matching. Using the graph representation for ontologies and schemas we proceed to rank the nodes of the graph using the lexical similarity of the ancestors. With the guiding intuition that, if the parent nodes match then their children are likely to match as well. This simple observation helps on to build a fast and efficient algorithm for matching different graphs (which represent ontology or schema). Since the algorithm is very fast it can be used as a quickly and dirty method to do initial matching of a large dataset and then drill down to the exact match with other algorithms. The algorithm is not dependent on the method used for calculating the lexical similarity so the best lexical analysis can be used to derive the node ranks. Once the node ranks are in place we can calculate the matching in just a single traversal of the graphs. No other algorithm that we know of can give such fast response time.*

Keywords: Ontology Matching, Schema Integration, Semantic Web

1 Introduction

An ontology is a specification of a conceptualization . It provides a vocabulary that describes a domain of interest and a specification of the meaning of terms used in the vocabulary. With the advent of the semantic web, ontologies have been used to describe what is expressed in computer format. They are viewed as the silver bullet for many applications, such as information integration, peer-to-peer systems, electronic commerce, semantic web services, social networks, and so on [1]. One of the major problem with ontologies can be that anyone can make any number of them, although efforts are being made to standardize them. Even with standards there can be ontologies for similar format derived from different sources. Ontology matching is one of the hot areas of concern to computer scientists all around the world. Several methods are already in place to enable the integration and match making for different ontologies [2]. Ontologies are usually represented as graph like structures (concept hierarchies, classifications, schemas). In this paper we present an algorithm based on ranking the nodes in the graph representation of the ontologies. This method as we shall see is a fast and efficient way to match ontologies.

The problem of ontology matching also pops up in a related field of schema integration from different heterogeneous databases. These schemas if represented in a graph form can also be matched using this algorithm. Several of well known algorithms in the semantic integration research have already been found beneficial for the database community [3]. We shall see here that node ranking can also be useful for schema integration. The schema can be conveniently represented as a graph with nodes as the attributes and relations as edges. Here is an example for a table 'Students'.

```
CREATE TABLE Students (sid CHAR (20),
                       name CHAR(20),
                       login CHAR(20),
                       age INTEGER,
                       gap REAL,
                       UNIQUE (name, age),
                       CONSTRAINT
StudentsKey PRIMARY KEY(sid))
```

This schema can be represented as a graph shown below.

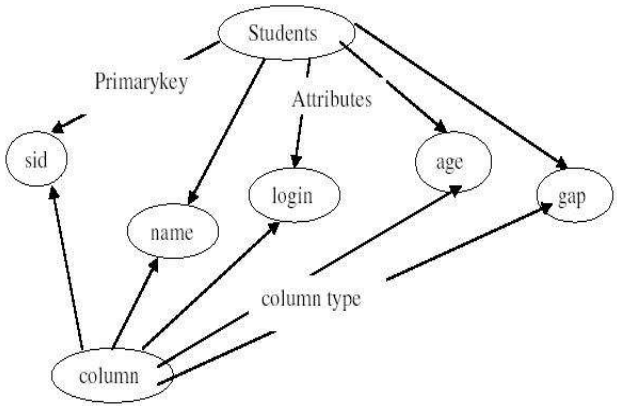

Figure 1: Graph representation of the schema

The attribute nodes can have further children if they refer to a foreign table and so on. This kind of graph can be used to match schemas using the given algorithm.

2 Node Ranking Algorithm

This algorithm makes use of the simple fact that if node A and A' match in an ontology represented as a graph then its likely that the nodes B and B' (the children) will also match.

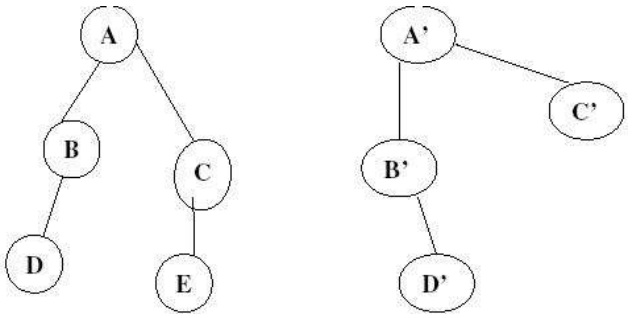

Figure 2: Representing ontologies

First of all we match the nodes with some lexical measure. Here we take the variation of the longest common subsequence algorithm to match the nodes according to their lexical similarity; this value is called the lex_sim of a node. The lex_sim of a node can have a value from 0 to 1 depending on the match. The following illustrates the algorithm followed to find out the lexical similarity. The strings X and Y are obtained from the nodes of the graph.

2.1 Algorithm Node Ranking:

Input: Strings X and Y with n and m elements, respectively

Output: For i=0,…,n-1,j=0,…,m-1,the length L[i,j] of a longest common subsequence of X[0..i] and Y[0..j]

```
for (i=-1 to n-1) do
        L [I,-1] = 0
for (j=0 to m-1) do
        L [-1, j] = 0
for (i=0 to n-1) do
        for(j=0 to m-1) do
        if (X[i]==Y[j]) then
                L[i, j] = L[i-1, j-1] + 1
        else
                L[i, j] = max{L[i-1, j],L[i, j-1]}
return array L
```

L[i,j] gives the length of the longest common subsequence, and from L[i, j] we can find out the lex_sim by dividing the length of X and Y by L[i, j]. So we have,

$$\text{lex\_sim(node A)} = m / L[i, j]$$
$$\text{lex\_sim(node A')} = n / L[i, j]$$

where m and n are the lengths of the strings A and A' respectively.

Now each node has a lex_sim measure with it, we take this lex_sim measure as the node's approval that its children and grand children will match as well. Thus in a way the lexical similarity of node is propagated to the children. Every node is associated with a node rank which is calculated as follows.

$$\text{node\_rank}(A) = \alpha * \text{lex\_sim}(A) + \beta * \text{lex\_sim}(\text{parent}(A)) + \gamma * \text{lex\_sim}(\text{grandparent}(A))$$

The values of the parameters α, β, γ are such that the node rank is a number between 0 and 1. The above equation calculates the node rank for a two generation propagation i.e. upto the grandparent level. Extending it to an 'n' ancestor node rank we get the following.

$$\text{node\_rank}(A) = \Sigma\, n_i * \text{lex\_sim}(\text{ancestor}_i(A))$$

The node_rank function is a normal distribution of the lexical similarity of the ancestors of that node. After the node ranks have been allotted the ontology graph looks like the one below.

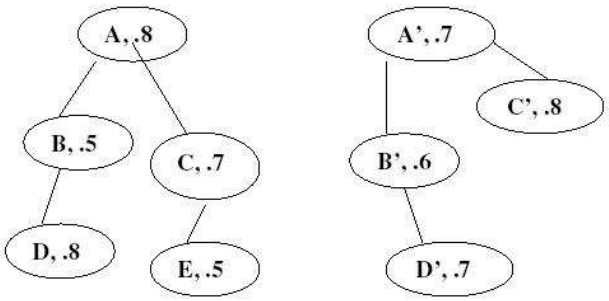

Figure 3: Ontologies after assigning node ranks

Once we have the node ranks of each node the matching problem can be tackled without any difficulty. Just making a traversal of the two graphs we match nodes with node ranks deferring in a small value say 'd'. This value may depend on the kind of ontology to be matched or the schema to be integrated. This method is similar to clustering the results by taking Manhattan distances. We can take on different values of 'd' for experimental data sets before deciding on the final value. Now we try to use this node ranking algorithm on the following ontologies of the departments.

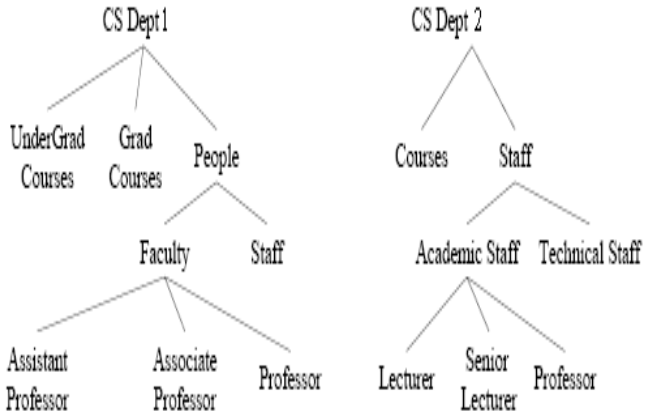

Figure 4: **CS department ontologies**

Applying the lexical matching algorithm for the various nodes we get the initial values by which w rank the nodes. The result of the initial lexical analysis is shown below.

Table 1: Lexical matching of the two ontologies

CS Dept 1 (0.86)	CS Dept 2 (0.86)
UG Courses (0.69)	Courses (1.0)
Grad Courses (0.63)	Courses (1.0)
Academic Staff (0.30)	Staff (1.0)
Technical Staff (0.38)	Staff (1.0)
Assistant Professor (0.5)	Professor (1.0)
Associate Professor (0.5)	Professor (1.0)
Professor (1.0)	Professor (1.0)

The lexical matching method can be improved on further by noticing the fact that when one of the words form a part of the other (has a lexical similarity of 1.0) , there is even more chance that they actually represent the same thing. Hence the values of several nodes like academic staff (with staff) can be increased by a certain level. Since the idea here is not to show the efficiency of the lexical analysis, we leave them as it is for the time been. Applying node ranking algorithm for two ancestors (i.e. parent and grand parent) the values at each node change to the following.

Table 2: After applying Node Ranking

CS Dept 1 (0.86)
UG Courses (0.69) + 0.2*0.86 = (0.808)
Grad Courses (0.63) + 0.2*0.69 = (0.768)
People 0.2*0.86 = (0.172)
Faculty 0.1*0.86 = (0.086)
Staff 0.1*0.86 = (0.086)
Assistant Professor (0.5) + 0.2*0.38 = (0.576)
Associate Professor (0.5) + 0.2*0.5 = (0.6)
Professor (1.0)
CS Dept 2 (0.86)
Courses (1.0)
Staff (1.0)
Academic Staff (0.30) + 0.2*0.63 = (0.426)
Technical Staff (0.38) + 0.2*0.30 = (0.456)
Lecturer 0.2*0.3 + 0.1*1.0 = (0.16)
Senior Lecturer 0.2*0.3 + 0.1*1.0 = (0.16)
Professor (1.0)

Now moving down the table choosing a value of 'd' as 0.4 and traversing the graph in top to bottom left to right we see that this algorithm matches perfectly the faculty and academic staff and then matches Assistant Professor <---> Lecturer , Associate Professor <---> Senior Lecturer, Professor <---> Professor and so on. These matching cannot be derived from the lexical analysis of the similarity measure along. Thus the node ranking gives a way of doing contextual analysis of the ontologies given the lexical similarities. In an exactly similar way we can go about matching the graphs representing schemas.

2.2 Algorithm Complexity Analysis

The complexity of the algorithm can be analyzed with respect to two things. One is the lexical matching algorithm and other is the node ranking. Since the node ranking algorithm is not dependent on the type of method chosen to do the lexical matching we can safely assume we have the lexical similarities with us. Then the next step is to construct a table similar to the Table 2. This requires that we traverse the given ontology graphs at least once. Several of the current known algorithms traverse the graphs many times to determine the proper matching [4], [5], and [6]. Once we have the table we need to make one more final traversal of the graph and cluster them as per the chosen 'd' values. This cluster represents the matching of the two graphs according to the ranks in their nodes.

3 Ontology Matching Results

This algorithm was tested on several ontologies taken from the DAML ontology library and several other ontologies created from dummy data sets. The accuracy of the algorithm was measured using the Longest Common Substring for the lexical matching and then using node ranking. The accuracy is defined with respect to the ability of the algorithm to successfully match more then 50% of the nodes in the given ontologies correctly. Here are the results.

Table 3: Matching results

Ontology	Matching
Publications	63%
Person	71%
Departments	83%
Courses	81%

The values in the matching column mean that the algorithm matched more then half of the nodes correctly, that many number of times for the given ontology. Say for courses the given ontologies matched 81% of times correctly (match is defined by matching more then 50% of nodes). The tests conducted involved choosing different values for 'd' and changing the number of ancestors up to which to apply the algorithm.

4 Future Work

The tests for various values of 'd' and choosing the right ancestor to apply the node raking are under way. And the results till now are promising. The tests for heterogeneous schemas are also going on. The following are the guidelines for future work.

- Testing the algorithm on a wider range of data sets.
- Testing the algorithm in conjunction with different lexical matching methods to find the one best suited.
- To use this algorithm in existing applications to complement and verify the results, those are already obtained.

5 Conclusion

In this paper we analyzed the node raking algorithm which gives a fast and time efficient way to match different ontologies and schemas. As the results suggest, this algorithm can be used as a quick and dirty method to match two ontologies. Since the algorithm fast in comparison to the ones used in current matching applications ([7], [8] and [9]) we can use it to do an initial survey of the datasets and identify the graphs that can later be analyzed in depth with other methods.

6 Acknowledgments

We would like to thank DARPA's Information Exploitation Office which sponsors DAML, which is the source (http://www.daml/org/ontologies) of all the datasets and ontologies used for testing the algorithm.

7 References

[1] D. Fensel *Ontologies: Silver Bullet for knowledge*, Management and Electronic Commerce, Springer-Verlag, 2001

[2] N. Noy: Semantic *Integration: A Survey of Ontology-based Approaches* Sigmod Record, Special Issue on Semantic Integration, 2004.

[3] A. Doan and A. Halevy: *Semantic Integration Research in the Database Community: A Brief Survey*. AI Magazine, Special Issue on Semantic Integration, 2005.

[4] R. Pan, Z. Ding, Y. Yu, Y. Peng: *A Bayesian Network Approach to Ontology Mapping*. Proceedings of ISWC, 2005.

[5] A. Gal*: Managing Uncertainty in Schema Matching with Top-K Schema Mappings.* Journal on Data Semantics, to appear, 2006.

[6] G. Stoilos, G. Stamou, S. Kollias: *A String Metric for Ontology Alignment.* Proceedings of ISWC, 2005.

[7] COMA++/COMA, http://dbs.uni-leipzig.de/Research/coma.html, University of Leipzig, Germany.

[8] CROSI, http://www.aktors.org/crosi/, University of Southampton/Hewlett Packard Laboratories, UK.

[9] MetaQuerier, http://metaquerier.cs.uiuc.edu/, University of Illinois at Urbana-Champaign, USA