F-023

Automatic Property and Its Value Weight Factors for Scalable Ontology Instance Matching

Rudra Pratap Deb Nath[*1] Md. Hanif Seddiqui[*2] Masaki Aono[*1]

1． Introduction

Ontologies, defined as “explicit formal specification of a shared conceptualization”, have become the backbone to enable the fulfillment of the semantic web vision. The rising demand of sharing data and digital resources within same or across organizations and heterogeneous sources has obtained a novel attention on the research issues of ontology alignment. However, ontology matching approaches and tools are gaining more and more significance in the framework of semantic web applications, where not only traditional matching at the schema level, but also and especially matching at the instance level is becoming essential to support discovery and management of different individual, with the introduction descriptions referring to the same real-world entity [1]. Ontology Instance Matching (OIM) compares different individuals within same or across heterogeneous knowledge bases to identify the same real world object. OIM problem has been widely investigated in several application domains where it is known with different names such as identity recognition, record linkage, entity resolution etc. OIM is also equally important into ontology population [2].

As the detection of mapping on schema level directly affect instance level matching, in this research, ontology schema matching and OIM work together. In schema level matching, our scalable anchor flood algorithm [4] is used. An instance can be described with the ontology concepts, properties and their values associated with it. However, every property doesn’t have equal contribution to identify an instance uniquely. Some properties have more influence in univocal instance identification. For example, in spite of having different values for the property “bankAccount” two individuals of type “Employee” are same if they have common values for “socialId”. So, automatically assigning a weight to each of the property is most important in OIM. Again, measuring the contribution of a property value in univocal instance identification is also more essential. Suppose, several individuals of “Person” type have contain common value namely ‘Hanif Seddiqui’ for the property “full_Name” while only one individual contains value ‘Masaki Aono’ for that property. Therefore, value ‘Masaki Aono’ should have more influence in matching process than value ‘Hanif Seddiqui’. Hence, it is also equally important to impose a weight automatically to each of the property values

Recently, several individual groups are also working to create billions of triples to represent ontology instances of semantic web which also raises the challenge of scalability in instance matching assignment. Here, we also introduce a naïve approach to make our instance matcher scalable by dividing the knowledge base into several smaller groups. Using owl disjointness relation, we produce separate clusters of related concepts from ontology. Then, only the instances of same category and same cluster of one ontology will be compared to the corresponding category and cluster of other ontology. The rest of the paper is organized as follows: Section 2 describes our instance matching approach. Section 3 outlines the experiment and evaluation. Final remarks and further scopes of improvement are discussed in section 4.

*1. Toyohashi University of Technology, Aichi, Japan.

*2. University of Chittagong, Chittagong, Bangladesh.

2. Instance Matching Algorithm

In this research, we augment our state of the art instance matcher [2] by computing similarity of two candidate instances based on weight factors of the properties and the property values related to instances. We also introduce a novel technique for addressing scalability issues.

2.1 Property and its Value Weighting

Automatically assigning more relevance to those properties that are considered as more significant for individual identification during the matching process is one of the key challenging issues in instance matching. Our basic hypothesis of weight factor measurement is that a property, for which instances contain more unique or distinct values, could have higher identification capability; “*the more the uniqueness, the more the weight factor*”. We define the weight factor of each property of an ontology concept C, used in a knowledgebase as follows:

$$Property\ weight\ factor\ (Wp\ )\ = \log|i \ni p^{u}|\ /\ \log|i| \qquad (1)$$

Where W_p is the property weight, $(|i \ni p^{u}|)$ is the number of instances where each contains unique value for property $p$and $|i|$ represents the number of instances of the concept *C*.

Again, measuring the contribution of a property value in univocal instance identification is also more essential. Property value weight factor rewards the value that has no repetition and penalizes those values that have more repetition. The weight factor of a property value is defined by following equation.

$$Property\ value\ weight\ factor\ (W_{pv_i})$$

$$= (1 -\ 1/(log|i \ni pv_i|) \qquad (2)$$

Where W_{pv_i} is the weight factor of the value v_i of property p. $|i \in pv_i|$ is the number of instances containing the value v_i for property p. Now, we consider both property level and value level weight factors in the affinity measurement of the SLCs [2] of two instances (in Eq.3). Semantic Link Cloud (SLC) defines an instance by all linked concepts, properties and their instantiations which are related to specify the instance sufficiently.

$$IA(slc_1\,, slc_2) = \frac{\sum_{p\,\epsilon\,\mathrm{slc}_1,\mathrm{q}\,\epsilon\,\mathrm{slc}_2}\left(\mathrm{E}(p,q)\,.(W_p{+}W_{pv}{+}W_q{+}W_{qv})\right)}{\sum_{p\,\epsilon slc_1}(W_p{+}W_{pv}) + \sum_{q\,\epsilon slc_2}(W_q{+}W_{qv})} \qquad (3)$$

E(p,q) returns 1 if string similarity of p and q is greater than a predefined threshold otherwise 0.

```
Algorithm:
instancMatch (ABox $ab_1$, ABox $ab_2$, Alignment A)
1. for each $ins_i \in ab_1$
2.   $slc_i$= generateSLC($ins_i$, $ab_1$)
3.     for each $ins_j \in ab_2$
4.       $slc_j$=generateSLC($ins_j$,$ab_2$)
5.         if $IA(slc_i,slc_j) \geq \delta$
6.           imatch=imatch ∪ makeAlign ($ins_i$, $ins_j$)
7.   return imatch
```

Fig 1. Instance Matching Algorithm.

Fig.1 describes a sample flow of the matching algorithm. Function generateSLC(*ins, ab*) collects an SLC against an instance from ABox *ab*.ABox contains all instances and their information of ontology.

2.2 Scalability in Instance Matching

Scalability is increasingly becoming an indispensable feature in instance matching. The motto of scalability is to accurately select a subset of instances that are more likely to be similar to an input instance, avoiding comparing the input instance against all the instances within the ontology. By using owl disjoint classes relation in an ontology, we produce a set of concept level clusters where each cluster includes related concepts by eliminating disjoin concepts and their descendants in ontology hierarchy. We use our anchor flood algorithm to get concept level cluster alignment. Then we apply our scalable instance matcher [2] only the concepts of corresponding cluster across ontologies. For each property of a concept, a category factor is defined by following equation.

$$Category\ factor\ (C_p) = log|i \ni p^u|/log|i \ni p| \qquad (4)$$

where ($|i \ni p^u|$) and ($|i \ni p|$) are the number of instances containing unique value and value for property p respectively. *candidatePropertySet* S_p of a concept is defined by including those properties whose have lower value than a defined threshold value. For two concepts across ontologies, we assign a *categoryPropertySet S* by taking the intersection of their *candidatePropertySets.*

```
Algorithm:
scalableInstancMatch (ABox ab1, ABox ab2, Alignment A)
1. for each a(b1,b2) ∈ A| b1∈cClusters(ont.1) ∧b2 ∈cClusters(ont. 2)
2 . for each (c1,c2) |c1∈ concept(ont.1∧ b1) ∧ c2∈ concept(ont.2∧ b2)
3.   if (catagoryPropertySet S ==null && |P1∩P2| !=0)
4.     IM=IM ∪instanceMatch (C1,C2, A)
5.   elseif (catagoryPropertySet S ==null && |P1∩P2|==0) continue
6.   else
7.     iCluster $s_1$= classifyInstances (c1, $ab_1$,categoryPropertySet S)
8.     iCluster $s_2$= classifyInstances (c2, $ab_2$ ,categoryPropertySet S)
9.       for each category cid
10.        if cid∋ ($iCluster_m$ ∈ iCluster $s_1$ ∧ $iCluster_n$ ∈ iCluster $s_2$)
11.          IM=IM ∪instanceMatch ($iCluster_m$ , $iCluster_n$ , A)
12. returnIM
```

Fig 2. Scalable Instance Matching (SIM) Algorithm.

Fig.2 shows the sample flow of SIM algorithm. For each aligned concept level clusters [at line 1], our SIM only efforts to compare instances of one concept of a cCluster against all instances of the concept of corresponding cCluster if their *categoryPropertySetS* has no member and both concepts have at least one common property [at lines 3 &4]. P1 and P2 indicate the property set of concepts *c1* and *c2* respectively.Concepts, those *S* has no element and have no common property will not compare the instances of each other [at line 5]. If the concepts' *S* have elements than the instances of both concepts will be categorized according to the values of *S*. For achieving more accuracy, we put the instances that contain null value for the S in another cluster. Now, only instances of same category [at line 10] will be compared by calling our previous instancMatch function.

3. Experiment and Evaluation

ISLab Instance Matching benchmark dataset are used for evaluation. It is a collection of OWL ontologies consisting of 29 concepts, 20 object properties, 12 data properties. IIMB 2010 is created by extracting data from Freebase, an open knowledgebase that contains about 11 million real world object containing information of movies, books, celebrations, company etc. Fig. 3 exhibits the performance of our KDE-SKEIM system over other methods attended in OM-2010 [3] workshop namely, ASMOV, CODI and RiMOM [3] in terms of recall-precision.

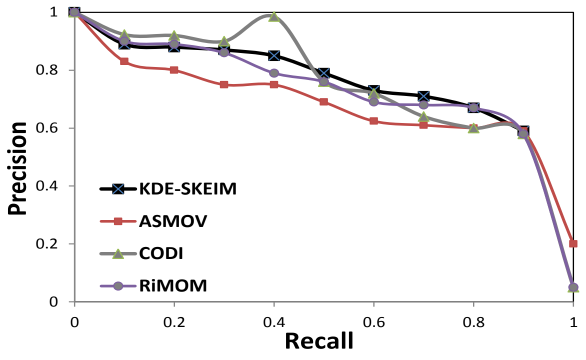

Fig 3. Recall-Precision Graph of Systems.

4. Conclusion

Automatic Property and Its value weighting provide better outcomes. Our scalable approach also boasts in terms of time complexity. We would like to evaluate our system with large repositories like DBpedia and DBLP. Our future plan also includes fitting the system with LOD (Linked Open Data) project.

References

[1]. S. Castano, A. Ferrara, S. Monanelli, and G. Varse , "*Ontology and Instance Matching, Knowledge driven multimedia information extraction and ontology evolution*" , LNCS 2011 volume 6050/2011.

[2] R. P. D. Nath, M. H. Seddiqui and M. Aono, "*An Efficient M-ethod for Ontology Instance Matching*", JSAI, Yamaguchi, 2012

[3] J. Euzenat, A. Ferrara, C.Meilicke et al "*Results of the Ontology Alignment Initiative*" Proceeding Ontology Matching 2010.

[4] M. H. Seddiqui and M. Aono, "*An Efficient and Scalable Algorithm for Segmented Alignment of Ontologies of Arbitrary Size.*" Journal of Web Semantics, Elsevier, page 344-356, 2009