1I2-R-4-1

# An Efficient Method for Ontology Instance Matching

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

[*1] Dept. of Computer Science & Engineering
Toyohashi University of Technology
Toyohashi, Aichi, Japan

[*2] Dept. of Computer Science & Engineering
University of Chittagong
Chittagong, Bangladesh

Ontology Instance Matching is a key interoperability enabler across heterogeneous data sources in the semantic web and a useful maneuver in some classical data integration tasks dealing with the semantic heterogeneous assignments. As heterogeneous sources of massive ontology instances grow dramatically day by day, scalability has become a major research issue in ontology instance matching of semantic knowledge bases. In this paper, we propose an efficient method by grouping instances into several sub-groups to address the scalability issue. Then, our instance matcher, which considers the semantic specification of properties associated to instances in matching strategy, works by comparing an instance within a classification group of one knowledge base against the instances of same sub-group of other knowledge base to achieve interoperability. A cunning approach for measuring the influence of properties in matching process is also presented. The experiment depicts satisfactory results in terms of effectiveness and scalability over baseline methods.

## 1. Introduction

Ontologies, defined as "explicit formal specification of a shared conceptualization [2]", have become the backbone to enable the fulfillment of the semantic web vision [6]. Nowadays, ontology alignment has been taken as a key technology to solve interoperability problems across heterogeneous data sources. It takes ontologies as input and determines as output an alignment, that is, a set of correspondences between the semantically related entities of those ontologies. These correspondences can be used for various tasks, such as ontology merging, data translation, query answering or navigation on the web of data. Thus, matching ontologies enables the knowledge and data expressed in the matched ontologies to interoperate. However, success of the vision of semantic web [1] depends on the availability of semantic linked data. Semantic linked data are all about connected data i.e. people, places and things are connected to each other. Each individual is known as instance [8]. With the development of Linked data and various social network websites, huge amount of semantic data is published on the web, which not only imposes new technology challenges over traditional schema level ontology alignment algorithms, but also demands new techniques for instance matching [7].

Ontology instance matching compares different individuals with the goal of identifying the same real world objects. In the literature, the instance matching problem has been widely investigated in several application domains where it is known with different names such as identity recognition, record linkage, and entity resolution problem etc. according to the requirements that need to be satisfied [8]. Instance matching plays a crucial role in semantic data integration as it interconnects all the islands of instances of semantic world to achieve the interoperability and information integration issues. Ontology instance matching is equally important in ontology population as it helps to correctly perform the insertion and update operation and to discover the relationship between the new incoming instance and the set of instance already stored in the ontology [10].

In several systems, [9,11] information of instances in an ontology is frequently used to support ontology schema matching. However, information of schema is equally importance in alignment of individuals that are sharing the same ontology structure. Moreover, several individual groups are also working to create billions of triples to represent ontology instances of semantic web which also raise the challenge of scalability in instance matching assignment. In this study, ontology schema matching and instance matching work together for discovering semantic mappings between possible distributed and heterogeneous semantic data. In [4], scalable ontology schema matching and in [5,8,12] instance matching are developed. However, further improvement of automatic weight generation and scalability in instance matching are still necessary which is the goal of this paper.

Our system uses our Anchor-flood algorithm to get the aligned concepts. According to the information of the instances of a concept, we automatically assign a weight factor, frequency factor and category factory to each of the properties of that class. Weight factor defines how many unique values a property contains. Frequency factor indicates how many instances contain values for the particular property. Category factor describes the efficiency of a property to be used for categorizing instances of a class. Then based on aligned concepts, we select common properties by which instances of both concepts are grouped. After that, our instance matcher, which considers the semantic specification of properties associated to instances in matching strategy, works by comparing an instances within a classification of group of one knowledge base (concept) against the instances of corresponding sub-group of other aligned concept.

The rest of the paper is organized as follows. Section 2 describes terminology frequently used throughout the paper. Our instance matching approach with automatic property weight factor is narrated in section 3. Scalability issue in instance matching depicts in section 4. Section 5 focuses on experiment and evaluation. Final remarks and further scopes of improvement are discussed in section 6.

Contact: Masaki Aono, Toyohashi University of Technology, 1-1 Hibarigaoka, Tempaku-cho, Toyohashi, Aichi, japan. 0532-44-6764, aono@tut.jp

## 2. General Information

This section introduces the basic definitions for familiarizing the readers with the notations and terminology used throughout the paper.

### 2.1 Ontology

An ontology is the basic element of the semantic web and the semantic knowledge base. According to M. Ehrig [3], an ontology contains a core ontology, logical mappings, a knowledge base, and a lexicon. Furthermore, a core ontology is defined as a tuple of five sets: concepts, concept hierarchy or taxonomy, properties, property hierarchy, and concept to property function.

$$S := (C, \leq_C, R, \sigma, \leq_R)$$

consisting of two disjoint sets $C$ and $R$ whose elements are called concepts and relations, two partial orders $\leq_C$ on $C$ called concept hierarchy or taxonomy and $\leq_R$ on $R$ called relation hierarchy, and a function $\sigma{:}R \longrightarrow C \times C$ called signature of binary relation where $\sigma(r) = (dom(r), ran(r))$ with $r \in R$, domain $dom(r)$ and range $ran(c)$. Ontology schema is often called as TBox.

### 2.2 Semantic knowledge base

The semantic knowledge base is a structure

$$KB := (C, R, I, \iota_C, \iota_R)$$

consisting of two disjoint sets $C$ and $R$ as defined be-fore, a set $I$ whose elements are called instances, two functions $\iota_C$ and $\iota_R$ called concept instantiation and relation instantiation respectively. ABox contains TBox-compliant statements about individuals belonging to those concepts.

### 2.3 Anchor-flood algorithm

Ontology instances are defined by the concepts and properties of ontology schema. Unless aligning schema entities, i.e. concepts and properties across ontologies, instance matching is not often achievable. Therefore, we use our scalable and efficient ontology alignment algorithm called Anchor-Flood to obtain alignment (a set of aligned-pair of schema entities, such as concepts, properties etc.) between ontology pair. Our Anchor-Flood algorithm achieved the best running time against the large ontology schema in Ontology Alignment Evaluation Initiative (OAEI), 2008 and 2009 [5].

Our scalable algorithm of ontology alignment starts off a seed point called an anchor (a pair of "look-alike" concepts from each of two ontologies). Starting off an anchor point, our scalable algorithm collects two sets of neighboring concepts across ontologies. Then it computes the structural and terminological similarity among the collected concepts and produces a list of aligned pairs. The collected concept pairs are in turn considered as further seed points or anchors. The operation cycle is repeated for each of the newly found aligned concept pairs. The cycle is stopped if there is no more new concept pair left to be considered as an anchor.

### 2.4 Semantic linked cloud (SLC)

In an ontology, neither a concept nor an instance comprises its full specification in its name or URI (Uniform Resource Identifier) alone. Therefore we consider the semantically linked information that includes concepts, properties and their values and other instances as well. They all together make an information cloud to specify the mean-ing of that particular instance. The degree of certainty is proportional to the number of semantic links associated to a particular instance by means of property values and other instances. The SLC is defined below:

A Semantic Link Cloud (SLC) of an instance is defined as a part of knowledge base [3] that includes all linked concepts, properties and their instantiations which are related to specify the instance sufficiently.

## 3. Instance matching approach

In this study, we improve our state of art instance matching algorithm by computing similarity of two candidates based on weight factors of the properties related to instances; i.e. each of the properties of a concept contains a weight factor. Moreover, for addressing scalability issue in instance matching we classify the instances of a concept by a specific method.

### 3.1 Property weight factor

Instances contain values associating with properties. Some properties may have higher influence than others in instance matching. For example, though both of instances have different values for the property *hasAge*, they are same individual if they have common values for the property *hasEmail* as the data is captured at different year. So, we can make as assumption that those properties have higher weight factors for which instances contain more unique values; i.e. "*the more the uniqueness, the more the weight factor*". Therefore, we define the weight factor of each property of an ontology concept used in a knowledgebase as follows:

$$weight\ factor, (\mathcal{P}_{\mathrm{w}}) = log(\,|\,\imath \ni \mathcal{P}_{\mathrm{u}}\,|\,) \,/\, log(|\imath|) \qquad (1)$$

where $\mathcal{P}_{\mathrm{w}}$ is the property weight, $\imath$ represents an instance, $(\,|\,\imath \ni \mathcal{P}_{\mathrm{u}}\,|\,)$ represents the number of instances contain the distinct or unique value for property $\mathcal{P}$, and $|\imath|$ represents the number of instances of the concept which has property $\mathcal{P}$ .

### 3.2 SLC generation

According to the definition of a Semantic Link Cloud (SLC), collection of linked information of an instance is an important step toward the instance matching. Collection of semantically linked resources of ABox along with concepts or properties of TBox specifies an instance at sufficient depth to identify instances even at a different location or with quite different label. Therefore, our proposed method collects all the linked information from a particular instance as a reference point. The linked information is defined as the concepts, properties or their values which have a direct relation to the reference instance [8, 12].

### 3.3 Instance matching algorithm (IMA)

Suppose two instances $\imath_1$ and $\imath_2$ of the same or aligned concept are given. The instance affinity function $\mathcal{IA}(\imath_1,\imath_2) \to [0,1]$ provides a value in the range [1,0]. $\mathcal{IA}$ is calculated by comparing the elements of SLCs of both instances. Here, we consider weight factors of properties in calculation.

Instances contain their lexical information as values of properties. String metric [13] is used for measuring similarity between two strings. Let Sp be a string and Sq be another string. The string based similarity between Sp and Sq is then measured as follows:

$$Sim(s_p, s_q) = com.(s_p, s_q) - diff.(s_p, s_q) + winkler(s_p, s_q)$$

where $com.(s_p, s_q)$ indicates the commonality between $s_p$ and $s_q$. $diff.(s_p, s_q)$ stands for the difference and $winkler(s_p, s_q)$ for the improvement of the result using the method introduced by Winkler [13].

Two property values $s_p$ and $s_q$ are equal if their string similarity is greater or equal to threshold. Therefore, we define an equality function $E_{pq}(s_p, s_q)$ as follows:

$$E(p,q) = \begin{cases} 1, & \text{if } \mathrm{Sim}(Sp, Sq) \geq \delta_1 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Let two instances *ins1* and *ins2* are represented by slc1 and slc2 respectively. So, affinity between ins1 and ins2 is measured by measuring the affinity between two SLCs; slc1 and slc2. We define the affinity between two SLCs by considering weight factor assigned to each of the property automatically as follows:

$$IA_{pf}(slc_1, slc_2) = \frac{\sum_{p \,\epsilon\, \mathrm{slc}_1, \mathrm{q} \,\epsilon\, \mathrm{slc}_2} \left(\mathrm{E}(p,q) \,.\, (\mathrm{W}_p + \mathrm{W}_q)\right)}{\sum_{p \,\epsilon\, slc_1} W_p + \sum_{\mathrm{q} \,\epsilon\, slc_2} W_q - \gamma} \tag{3}$$

where γ indicates the factors for missing property values.

```
Algorithm: instancMatch (ABox ab1, ABox ab2, 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 $\exists$ a($c_1$,$c_2$) $\in$ A|$c_1 \subseteq$ concept($ins_i$) $\wedge$ $c_2 \subseteq$ concept($ins_j$)
6.          if IA(slci,slcj)$\geq \delta_2$
7.              imatch=imatch $\cup$ makeAlign ($ins_i$,$ins_j$)
```

Figure1. Pseudo code of Instance Matching Algorithm.

Fig.1 describes a simple flow of the matching algorithm. For an SLC of an instance is matched against every SLCs of instances of knowledge base (line 1 through 4 in Fig. 1) if and only if there concepts are aligned (as there exists a condition at line 5 in Fig. 1). Function generateSLC(*ins,ab*) collects an SLC against an instance *ins* in ABox *ab*. An SLC usually contains concepts, properties, and their consolidated values. Every value of an SLC is compared with that of another SLC (as of line 6 of Fig. 1). Once similarity value is greater than the threshold, it is collected as an aligned pair (as stated at line 7 in Fig. 1). Finally, the algorithm produces a list of matched instance pairs.

## 4. Scalability issue in Instance matching

Day by day, the size of knowledge bases is increasing sharply. For example, even older *dblp* contains 400,000 authors while there are 198,056 persons in *Dbpedia*. Our brute force algorithm compares every author of *dblp* to every persons of Dbpedia (as *person* is aligned with *author*). So, it requires 400,000*198,056 SLCs comparisons. Hence, it suffers from scalability problem. Here we propose an efficient method by automatically further classifying the instance set of a concept into several sub-groups according to special properties that are selected by analyzing their property category factor. Following sub-sections describe in details.

### 4.1 Property frequency factor

For each property of a concept, property frequency factor is assigned. Property frequency factor describes how many instances contain values for that property. So, that property has more frequency factor for which more instances contain values for it. For example, *name* property of *person* concept may have high frequency factor as every person must have a name. We define the frequency factor of a particular property by following equation.

$$\textit{frequeny factor}, (\mathcal{P}_f) = log(\mid \acute{\imath}\ \text{з}\ \mathcal{P} \mid) \,/\, log(|\acute{\imath}|) \tag{4}$$

where $\mathcal{P}_f$ is the property weight, $\acute{\imath}$ represents an instance, $(\,|\acute{\imath}\ \text{з}\ \mathcal{P}|\,)$ represents the number of instances contain values for property $\mathcal{P}$, and $|\acute{\imath}|$ represents the number of instances of the concept that has property $\mathcal{P}$ .

### 4.2 Property Category factor

Category factor of a property defines how much it is appropriate for grouping the instances of the concept. We define category factor of property in a cunning way so that the properties which have little weight factor but high frequency factor become better candidates for instance categorization. Equation 5 defines the property category factor mathematically.

$$\begin{aligned} \textit{category factor}, (\mathcal{P}_c) &= \mathcal{P}_w / \mathcal{P}_f \\ &= \log_{(\,|\,\mathrm{i}\ \text{з}\ \mathcal{P}\,)|} (|\,\mathrm{i}\ \text{з}\ p_u\ |) \end{aligned} \tag{5}$$

The lower value for category factor of a property indicates better candidate.

### 4.3 Selection of properties for grouping instances

We set a threshold value for choosing properties from available properties of concept. That is a property is a member of the set of candidate property, $S_p$ if its category factory is lower than the threshold value. Now, we finalize the properties selection by taking the intersection of the sets of candidate properties of aligned concepts. For example, concept $C_1$ and concept $C_2$ are aligned and the set of candidate property of $C_1$ and $C_2$ are $C_1 s_p$ and $C_2 s_p$ respectively. So, the final set of property S, by which instances of both concepts will be grouped, is

$$S = C_1 s_p \cap C_2 s_p \tag{6}$$

### 4.4 Scalable instance matching algorithm

```
Algorithm: scalableInstancMatch (ABox ab1, ABox ab2, Alignment A)
1. for each a(c1,c2) $\in$ A |c1$\in$ concept(ont.1) $\wedge$ c2 $\in$ concept(ont.2)
2.    cluster $s_1$= classifyInstances (c1, $ab_1$, selectedPropertySet S)
3.    cluster $s_2$= classifyInstances (c2, $ab_2$, selectedPropertySet S)
4.        for each Category cid
5.          if cid $\ni$ ($cluster_m$ $\in$ cluster $s_1$ $\wedge$ $cluster_n$ $\in$ cluster $s_2$
6.             call IM=IM $\cup$ instanceMatch ($cluster_m$, $cluster_n$, A)
7.                return IM
```

Figure 2. Scalable Instance Matching Algorithm

Fig. 2 describes our propose scalable algorithm for matching instances of large knowledge bases which works for each aligned concepts pair given by our ontology schema matching Anchor-flood algorithm (sec.2.3). Function classifyInstances classifies the instances of a concept according to the selected properties values. For a pair of common category we call our previous algorithm instanceMatch (Fig.1). Finally, algorithm returns the set of aligned instance pairs.

## 5. Experiment and Evaluation

ISLab Instance Matching Benchmark is used for evaluation. The test-bed provides OWL/RDF data about actors, sport persons, and business firms. The main directory contains 37 sub-directories and the original ABox and the associated TBox (abox.owl and tbox.owl). Each sub-directory contains a modified ABox (abox.owl + tbox.owl) and the corresponding mapping with the instances in the original ABox (refalign.rdf). The benchmark data is divided into four major groups: value transformation (001-010), structural transformation (011-019), logical transformation (020-029) and combination transformation (030-037).

OAEI-2009 introduces instance matching track to the participants. The following recall-precision graph (Fig.3) depicts the performances of different systems where our Anchor-flood (*AFlood*) algorithm works without considering the weight factors of properties. Fig. 4 describes the improvement of our proposed method over different transformation in comparison with previous one. Here, we use the well known precision and recall as measurement matrices. Moreover, our proposed one boasts for less computational time.

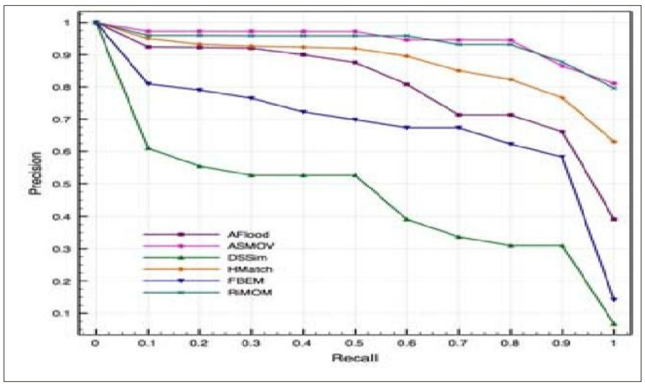

Figure 3: Instance Matching Results in OAEI-2009 Campaign

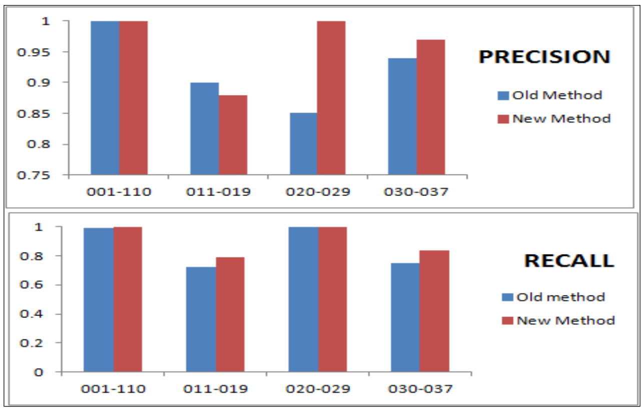

Figure 4: Comparison of proposed method with previous one

## 6. Conclusion

An efficient method of assigning automatic weight to each of the property and addressing scalability issue to improve our state-of–the-art ontology instance matching is presented. As matching of instances are computing by considering the weight factors of the properties associated to them, better outcomes in terms of precision and recall are achieved. Though our scalable algorithm takes some extra times for pre-processing, it reduces the SLCs comparisons sufficiently. However, precision is also a major concern when the sizes of datasets are increasing as our algorithm matches the instances only if their concepts are aligned. We will try to achieve further improvement by addressing the same individual that resides in non-aligned concepts across knowledge bases. Testing our algorithm with large knowledge bases like Dbpedia and dblp as well as to fit it with LOD (Linked Open Data) project are also our future concerns.

## References

[1] T. Berners-Lee, M. Fischetti, M. Dertouzos, Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web, Harper San Francisco, 1999.

[2] R. Studer, V. Benjamins, D. Fensel, Knowledge Engineering: Principles and Methods, Journal of Data & Knowledge Engineering 25 (1-2) (1998) 161–197.

[3] M. Ehrig, Ontology Alignment: Bridging the Semantic Gap, Springer, New York, 2007.

[4] M. H. Seddiqui, M. Aono, An Efficient and Scalable Algorithm for Segmented Alignment of Ontologies of Arbitrary Size, Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 2009

[5] M. H. Seddiqui, M. Aono, Anchor-Flood: Results for OAEI-2009, Proceedings of Ontology Matching Workshop of the 8th International Semantic Web Conference, Chantilly, VA, USA, 2009

[6] J. Tang, J. Li, B. Liang, X. Huang, Y. Li, K. Wang, Using Bayesian decision for ontology mapping, Journal of Web Semantics: Science, Services & Agents on the World Wide Web, 2006

[7] Z. Wang, X. Zhang, L. Hou, Y. Zhao, J. Li, Y. Qi, J. Jang, RiMOM Results for OAEI 2010, Proceedings of Ontology Matching Workshop of the 9th International Semantic Web Conference, Shanghai, china, 2010

[8] M. H. Seddiqui, S. Das, I. Ahmed, R.P.D. Nath, M. Aono, Augmentation of Ontology Instance Matching by Automatic Weight Generation, World congress on Communication and Technology, India, IEEE, 2011

[9] A. Isaac, L. V. D. Meij, S. Schlobach, S. Wang, An Empirical Study of Instance Based Ontology Matching, ISWC/ASWC 2007

[10] S. Castano, A. Ferrara, D. Lorusso, S. Montanelli, On the Ontology Instance matching Problem, 19th International Conference on Database and Expert Systems Application , Italy, IEEE, 2008

[11] K. K. Breitman, D. Brauner, M. Casanova, A. Perazolo, Instance-Based Ontology Mapping, Fifth IEEE Workshop on Engineering of Autonomic and Autonomous systems, IEEE, 2008

[12] M. H. Seddiqui, M. Aono, Ontology Instance Matching by considering Semantic Link Cloud, Proceedings of 9th WSEAS International Conf. on Applications of Computer Engineering.2010

[13] W. E. Winkler, The State of Record Linkage and Current Research Problems, Technical report, Statistical Research Division, U.S. Census Bureau, Washington DC, 1999.