Discovery of similar blocks from very large-scale ontologies

Aicha Boubekeur[1], Abdellah Chouarfia[2]

[1] Computer Science Department, University of Tiaret, Algeria,
boubakeur@univ-tiaret.dz
[2] Computer Science Department, University of UST-Oran, Algeria,
chouarfia@univ-usto.dz

Abstract. Large scale ontology matching is a labour-intensive and time-consuming process. To alleviate the problem, many automated solutions have been proposed. In order to avoid the drawbacks of the existing solutions, this paper proposes to cut down the number of concept pairs for which a similarity measure must be computed during ontology matching. More important, the main contribution is to deal subsets of concepts pair: on the one hand, if two concepts are highly similar, we leverage the concept hierarchy to skip subsequent matching between sub-concepts of one concept and super-concepts of the other concept. On the other hand, if two concepts are weakly similar, we leverage the locality phenomenon of matching to skip subsequent matching between one concept and the neighbours of the other concept.

Keywords. Large ontology, neighbour, similarity, link, search space.

1 Introduction and Motivation

In recent years, many large ontologies are created and maintained in the areas including machine translation, information retrieval, e-commerce, digital library, medicine, and life science. These ontologies have more than thousands to millions of concepts and properties, and some of them contain more than billions of instances such as Cyc[1], WordNet[2], SUMO[3], Gene Ontology[4] and UMLS[5].
It has been argued that the difficulties of the operations of constructing, matching, reusing, maintaining, and reasoning on large ontologies would be extremely simplified by splitting large ontologies into smaller modules which cover specific subjects [6, 8]. Ontology modularization is the collective name of two approaches for frag-

[1] http://www.cyc.com/

[2] http://wordnet.princeton.edu/
[3] http://www.ontologyportal.org/
[4] http://www.geneontology.org/
[5] http://www.nlm.nih.gov/research/umls/

menting ontologies into smaller, coherent components (modules), which are themselves ontologies [5]:

Ontology Partitioning Approaches. The ontology is partitioned into a number of modules $\{M_1, \cdots, M_n\}$ such that the union of all the modules is semantically equivalent to the original ontology $\{M_1 \cup M_2 \cup ... \cup M_n\} = O$; i.e. the M_i modules are not necessarily disjoint. Thus, a function partition (O) can be formally defined as follows:

$$\textbf{\textit{Partition}}\ (O) \longrightarrow M = \{\{M_1, M2, ..., Mn\} \mid \{M_1 \cup M_2 \cup ... \cup M_n\} = O\} \qquad (1)$$

The partitioning method reduces the search space and thus leads to better efficiency. The space complexity of the matching process is also reduced. Four partition based methods COMA++ [2], FalconAO [7,3], Taxomap [3], anchor flood [7] will be discussed below.

Module Extraction techniques. Concepts that form a coherent fragment of an ontology O are extracted to form a module M, such that it covers a given vocabulary (based on an initial module signature) Sig(M), Such that Sig(M ⊆ Sig(O) [9]. In fact this task consists in reducing an ontology to the sub-part, the module, that covers a particular sub-vocabulary of O, as such M ⊆ [9]. Note that M is now ontology itself. A function extract (O, Sig (M)) can be defined as follows:

$$\textbf{\textit{Extract}}\ (O, Sig\ (M)) \longrightarrow \{M \mid M \subseteq O\} \qquad (2)$$

There are numerous techniques [4, 5] for module extraction, more than ontology partitioning approaches that have been developed for different purposes. The main usage of these approaches concerns partial reusing, when an application or a system needs only a part of ontology. Broadly speaking, modularization approaches aim to identify the minimal set of necessary concepts and definitions for different parts of the original ontology.
However, ontology partitioning approaches present several drawbacks. They cannot control the size of blocks, which may be too small or too large for matching [3, 4, 5, 6,9]. They can also cause another problem, namely, the partitioning can make the elements on the boundaries of blocks lose some semantic information, which in turn affects the quality of final matching results. This paper proposes a generic solution to assess preliminary 1: n mappings between any two concepts from two given ontologies based on their descriptive (semantic) information. On the one hand, if two concepts are highly similar, we leverage the concept hierarchy to skip subsequent matching between sub-concepts of one concept and super-concepts of the other concept. On the other hand, if two concepts are weakly similar, we leverage the locality phenomenon of matching to skip subsequent matching between one concept and the neighbors of the other concept.

The paper is structured as follows: Section 2 discusses large scale matching techniques. Section 3 presents definitions and basic concepts used throughout the paper. Section 4 describes our structure-based matching approach. Finally, Section 5 provides some concluding remarks.

2 Related work

According to Shvaiko P and Euzenat J [1] one of the toughest challenges for matching system is handling large scale schemas or ontology. Large-scale ontologies are a kind of ontologies created to describe complex real world domains. So, various large scale matching techniques are categorized in [2]:

- The early pruning strategy is to reduce the search space for matching; one matcher can prune entity pairs whose semantic correspondence value is very low, thus reducing search space for the subsequent matcher (Quick Ontology Matching algorithm (QOM), Eric peukert et al. schema and ontology matching algorithm).

- The partition strategy is performed in such a way that each partition of first ontology is matched with only small subset of the partitions of the second ontology. This method reduces the search space and thus better efficiency (Coma++, Falcon-AO, Taxomap and Anchor flood).

- The parallel matching technique has two kinds' inter-matcher and intra-matcher parallelization. Inter-matcher parallelization deals with parallel execution of independently executable matchers while intra-matcher parallelization deals with internal decomposition of individual matchers or matcher parts into several match tasks that can be executed in parallel (Gross & al. ontology matching algorithm).

- Other matching tool: RiMOM and ASMOV ontology matching tools, Agreementmaker schema and ontology matching tool.

3 Preliminaries

The following definitions and basic concepts are used throughout the paper:

Definition 1 *(*schema graph*)*: A schema graph (directed acyclic graph) of an ontology is given by (V,E,Lab_v), where: V = {r, v_2, ..., v_n} is a finite set of nodes, each of them is uniquely identified by an object identifier (OID), where r is the schema graph root node. E = {(v_i, v_j)|v_i, v_j ∈ V } is a finite set of edges. Lab_v is a finite set of node labels. These labels are strings for describing the properties of the element and attribute nodes, such as name and data type.

Definition 2 (neighbor): A neighbor concept c can be defined as follows: Neighbors(c) = {Sub(c) U Sup(c)} avec Sub(c) = {c'| c' sub-concept c} and Sup(c) = {c'| c' sup-concept c}

Definition 3 *(*strong-Links*):* Given two schema graph G=(O1, E, Lab_v) and G'=(O2, E', $Lab_{v'}$) of ontologies O1 and O2, the similarity values between a_i∈S and concepts b_1, b_2, …, b_n in ontology O2 are Sim (a $_i$)={ sim (a_i, b_j)∈ G X G' | j=1..n}, and the strong-Links of a node a_i∈O1 is given by $S_N(a_i)$= { *bj* | sim (a_i, b_j) ≥ thresold}, thresold is a high value in [0..1].

Definition 4 *(*low-Links)*:* Given two schema graph G= (O1, E, Lab_v) and G'= (O2, E', $Lab_{v'}$) of ontologies, the similarity values between a_i ∈O1 and concepts b_1, b_2, …, b_n in ontology O2 are Sim (a $_i$)={ sim (a_i, b_j) ∈ G X G' | j=1..n}, and the low-Links of a node a_i∈O1 is given by $L_N(a_i)$={*bj* | sim (a_i, b_j) < thresold}, thresold is usually a small value in [0..1].

Through these two last definitions, the matching process can reduce maximum times of similarity computation and thus reduce the time complexity significantly.

4 Structure connected Links

Our structure-based matching approach is realized by:

Step1. This phase is concerned with the representation of heterogeneous ontologies as sequence representations. First, each ontology is parsed and represented internally as a rooted ordered labeled graph, wherein each graph component (element and/or attribute) is represented as a node, while edges are used to represent relationships between components. Each node in the schema graph carries the associated element properties.

Step2. Compute preliminary similarities between any two entities for two given ontologies based on their descriptive information i.e. generate set of concepts pairs or links. It utilizes both structural and linguistic information for initial alignment and then applied subsequent similarity propagation strategy to produce more alignments if necessary. It main function is to match the heterogeneous ontologies.

Step3. The first issue is to extract two kinds of virtual sub-graph for highly / weakly similar concepts (links) across ontologies. The second issue is to reduce the search space (i.e. Space and time complexity of the matching process), concerning wide-scale semantic heterogeneity in matching: this phase specifies all the similarity to be computed, and among these calculations, several links can be skipped in matching process.

Step4. During matching process, if credible alignments are computed, the corresponding high similarity links are isolated. Such links are to predict the ignorable similarity calculations in the remaining matching process. Also if the incredible matching results are found, the corresponding negative reduction' Links according to the locality of matching are also constructed, and such links to predict the ignorable similarity calculations are utilized. The similarity measure between entities from the two ontologies is computed by analyzing the literal and structural information in semantic subgraph extract in previous part.

Step 5. Repeat the two last steps for more alignment.

To this end, this process aims at providing high quality alignments between concept pairs with a time processing limit reasonable and it not needs to modularize or partition the large ontologies.

Therefore, considering structural information is a natural way for enhancing ontology mapping as illustrated by: Given two entities *ai* from O1 and *bj* from O2, we first apply and compute the similarities between entities based on the similarities of words e.g. the string-based and WordNet-based methods:

String-based method. the similarity measure between words w_i and w_j is defined as:

$$simStr(wi,wj) = comm(wi,wj) - diff(wi,wj) + winkler(wi,wj) \qquad (3)$$

where *comm(wi,wj)* stands for the commonality between *wi* and *wj* , *diff(wi,wj)* for the difference between *wi* and *wj* , and winkler (*wi,wj*) for the improvement of the result using the method introduced by Winkler in [7].

WordNet-based method. We use an electronic lexicon, WordNet, for calculating the similarity values between words. The similarity between two words *wi* and *wj* is measured by using the inverse of the sum length of the shortest paths [6]:

$$sim_{WN}(wi,wj) = 1 / (llength + rlength) \qquad (4)$$

Where *llength* is the shortest path from word node *wi* to its common hypernym with word node *wj* and *rlength* denotes the shortest path from *wj* to its common hypernym with *wi*.

Instead of matching to all concepts by traversing taxonomies completely, the goal is to find Links between ontologies, at this step, it only considers on finding Links from the Cartesian product (X) of the two ontologies. These Links, are very important matching concepts, are used to reduce the time complexity in matching without exploring other commonalities between neighbors from the corresponding Links (initial Links generation). The algorithm proposed here generates a set of matching concepts as the initial links (see Algorithm1). The function *Sim* is an aggregated similarity function incorporating name and structural similarities (step 2):

Algorithme 1:

```
Input: Two ontologies O1,O2
Output: Neighbor-set
For each pair (ai,bj) $\in$ O1xO2 do
  Compute sim(ai,aj)
  If (sim (ai,bj) > 0)
     then Links $\longleftarrow$ U {(ai,bj)}
  End
  Return (Neighbor-set)
End
```

Algorithme 2:

```
Input: Ontology O1, Ontology O2, Links
Output: Set of Strong-Links
   Links are generated by algorithme1
   Getstrong-Links $\Longleftarrow$ ai
   SN $\longleftarrow$ Ø
   For each bj $\in$ O2 do
     Compute sim(ai,bj)
     If sim(ai,bj) > threshold
              then SN $\longleftarrow$ U {bj}
     End
   End
   Return Getstrong-Links
```

For finding efficient results, two possibly solutions are provided:

- If concept A matches concept B, it needs not to calculate the similarity between sub-concepts (/super-concepts) of A and super-concepts (/sub-concepts) of B, thus we can reduce the total times of similarity calculations.
- If A does not match B, it is very possible that their neighbors also do not match each other that imply we can ignore many similarity calculations.

Obviously, it needs to discover the high-Links and the low-Links dynamically in matching, and then uses these Links to optimize similarity calculations. For $S_N(a_i)=\{b_1, b_2,\ldots,b_n\}$, the strong-Links set RS_N (a_i) is calculated by:

$$RS_N\ (a_i)=\bigcap_{j=1}^{k} RS_N\ (a_i|b_j)=[sub(a_i)Xsup(lub(b_1,\ldots, b_k))]\ U\ [sup(a_i)Xsub(glb(b_1,\ldots, b_k))]$$

With $lub(b_1,\ldots, b_k)$ and $glb(b_1,\ldots, b_k)$ are the least upper bound and the greatest lower bound for $(b_1,\ldots, b_k)$. Apparently, the total strong-Links sets during the matching process is $RS_N = U\ RS_N\ (a_i)_{\ i=1,n}$ (see Algorithm2 & 3):

Algorithme 3:

```
Input: Ontology O1, Ontology O2, Strong-Links
Output: total strong-Links sets
   StrongLinks are generated by algorithme2
   Matchedset $\Longleftarrow$ strong-Links ($a_i$)
   Generates the neighbors of $a_i$        {sub($a_i$) | sup($a_j$)}
 For each $b_j \in S_N$
     Generates the neighbors of $b_j$        {sub($b_j$) | sup($b_j$)}
```

```
    $RS_N \leftarrow$ U {[sub($a_i$) X sup(lub($b_1,\ldots, b_k$))] U [sup($a_i$) X sub(glb($b_1,\ldots, b_k$))] }
**End**
**Return** Matchedset
```

5 Conclusion

First of all, the analysis in the existing matching systems depicts that there is always a tradeoff between effectiveness and efficiency. The main goal of this paper is to deal with wide-scale semantic heterogeneity in large scale ontology matching. For this purpose, we focus on reducing complexity, concerning wide-scale semantic heterogeneity in space matching. To accomplish this, we propose to skip subsequent matching between sub-concepts of one concept and super-concepts of the other concept (of shortcuts) of ontologies as input. However, it may be asked if this solution is quite adapted to find the most correct mappings between two concepts and the offline discovering mappings from different ontologies. As a future work, we aim at answering these questions.

6 References

1. Shvaiko P, Euzenat J, “Ten challenges for ontology matching,” *Confederated International Conference on the Move to Meaningful Internet Systems*, pp. 1164–1182,2008.
2. Rahm E, “Towards Large-Scale Schema and Ontology Matching,” *Schema matching and mapping*, Bellahsene Z, Bonifati A Rahm E, eds. New York: Springer Heidelberg, pp. 3–27, 2011.
3. F. Hamdi, B. Safar, C. Reynaud, and H. Zargayouna. Alignment-based partitioning of large-scale ontologies. In Advances in Knowledge Discovery and Management, volume 292, pages 251–269. Springer, 2010.
4. J. Seidenberg and A.L. Rector, “Web ontology segmentation: Analysis, classification and use”, In *Modular Ontologies*, H. Stuckenschmidt, C. Parent and S. Spaccapietra, LNCS 5445, Springer, 2009, pp. 211–243.
5. Doran, Paul *Ontology modularization: principles and practice*. Doctoral thesis, University of Liverpool , octobre (2009) .
6. P. Bouquet, L. Sera¯ni, S. Zanobini: Semantic coordination: A new approach and an application. In Proceedings of the 2nd Int. Semantic Web Conf. (ISWC'03). (2003) 130-145
7. G. Stoilos, G.B. Stamou, S.D. Kollias: A string metric for ontology alignment. In Proceedings of the 4th Int. Semantic Web Conference (ISWC'05) (ISWC'05). (2005) 624-637
8. I. Palmisano, V. Tamma, T. Payne and P. Doran, “Task oriented evaluation of module extraction techniques”, In *ISWC*, LNCS 5823, Springer, 2009, pp. 130–145.
9. M. d’Aquin, A. Schlicht, H. Stuckenschmidt and M. Sabou, “Criteria and evaluation for ontology modularization techniques”, In *Modular Ontologies*, H. Stuckenschmidt, C. Parent and S. Spaccapietra, LNCS 5445, Springer, 2009, pp. 67–89.