Matching Large Ontologies Based on Reduction Anchors

Peng Wang [1], **Yuming Zhou** [2], **Baowen Xu** [2]
[1]School of Computer Science and Engineering, Southeast University, China
[2]State Key Laboratory for Novel Software Technology, Nanjing University, China
pwang@seu.edu.cn, zhouyuming@nju.edu.cn, bwxu@nju.edu.cn

Abstract

Matching large ontologies is a challenge due to the high time complexity. This paper proposes a new matching method for large ontologies based on reduction anchors. This method has a distinct advantage over the divide-and-conquer methods because it dose not need to partition large ontologies. In particular, two kinds of reduction anchors, positive and negative reduction anchors, are proposed to reduce the time complexity in matching. Positive reduction anchors use the concept hierarchy to predict the ignorable similarity calculations. Negative reduction anchors use the locality of matching to predict the ignorable similarity calculations. Our experimental results on the real world data sets show that the proposed method is efficient for matching large ontologies.

1 Introduction

Recent years have seen an increasing use of large ontologies in various areas such as machine translation, e-commerce, digital library, and life science. Since building and maintaining large ontologies may be distributed and autonomous, large ontologies can also be heterogeneous. Ontology matching is a plausible solution to the heterogeneity problem. However, large ontology matching (LOM) is a challenge due to the high time complexity and space complexity. First, matching process requires a large amount of memory. This may cause the matching system to crash by the out of memory error. Second, most LOM methods are $O(n^2 \times t)$ time complexity (n represents the number of concepts), i.e. it needs n^2 times similarity calculations and each similarity calculation has $O(t)$ complexity. This paper focuses on the time complexity in the LOM problem.

Most existing ontology matching systems are unable to deal with the LOM problem. The Ontology Alignment Evaluation Initiative (OAEI[1]) results in past years showed that few systems could deal with LOM tasks. In OAEI2007, only 2 of all 18 systems finished 4 LOM tasks: $anatomy$, $food$, $environment$, and $library$. In OAEI2008, only 2 of all 13 systems finished 4 LOM tasks: $anatomy$, fao, $mldirectory$, and $library$, and only 1 system finished the $vlcr$ task.

[1]http://oaei.ontologymatching.org/

Divide-and-conquer strategy is a feasible solution to reduce the time complexity in LOM by partitioning a large ontology into small modules. However, it has two limitations. First, most existing ontology partitioning approaches cannot control the size of modules [Hu and Qu2006, Hu *et al.*2008]. Consequently, many too small or too large modules, which are inappropriate for matching, may be generated. Second, partitioning ontologies into modules may lead to the loss of useful semantic information on the boundary elements. As a result, the quality of ontology matching may be degraded.

In this paper, we propose a reduction anchors based approach for matching large ontologies. Compared to existing work, the proposed approach has the following advantages. First, it does not need to partition ontologies but keeps the high performance as the divide-and-conquer approaches have. Second, it is indeed a general LOM framework, in which most existing matching techniques could be used. The main contribution of this paper is that we introduce two types of reduction anchors to cut down the number of pairs for which a similarity measure must be computed during ontology matching. On the one hand, if two concepts have a high similarity, 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 have a low similarity, we leverage the locality phenomenon of matching to skip subsequent matching between one concept and the neighbors of the other concept. The former is called a positive reduction anchor and the latter is called a negative reduction anchor. Our experimental results show that the proposed approach is very effective for matching large ontologies.

2 Related Work

The LOM problem has been concerned by both academic researchers and industrial engineers. For example, people integrated common large ontologies for machine translation [Hovy1998], discovered mappings between Web directories for information retrieval [Massmann and Rahm2008], and matched biology and medical ontologies [Zhang *et al.*2007, Mork and Bernstein2004]. This paper classifies existing LOM solutions into three types: quick-similarity-calculation (QSC) methods, parallel processing (PP) methods, and divide-and-conquer (DC) methods.

QSC methods attempt to reduce the time complexity in each similarity calculation, namely, the factor t in $O(n^2 \times t)$. To this end, they often use simple but quick matchers such as literal-based and structure-based matcher. However, previous literature shows that, when matching large ontologies,

QSC methods have a high time complexity [Mork and Bernstein2004]. In OAEI2007, some systems with QSC methods had not any advantages both in running time and the quality of matching results. Indeed, QSC methods are unable to deal with LOM for the following two reasons. First, quick matchers only use limited information that would cause low-quality results. Second, since n^2 is a large number, reducing factor t has little influence on the matching performance.

PP methods employ the parallel strategy to deal with the similarity calculation [Mao2008]. The parallel processing idea is very simple and easy to be implemented. However, it needs expensive hardware resources to set up the parallel computing environment. More importantly, the matching performance improvement is limited.

DC methods attempt to reduce the factor n^2 in $O(n^2 \times t)$. The divide-and-conquer strategy partitions a large ontology into k modules or blocks to reduce the time complexity to $O(\frac{n^2}{k} \times t)$. The improvement of performance is determined by the number of modules. Modular ontology is a popular way to partition large ontologies. However, existing modular ontology methods focus on the correctness and completeness of logics but cannot control the size of modules [Hu *et al.*2008], i.e., they would generate too large or too small modules. For example, a modularization algorithm will generate the large module with 15254 concepts for NCI ontology and will fail for GALEN ontology [Grau *et al.*2007].

Malasco [Paulheim2008] and Falcon-AO [Hu *et al.*2008] are two well-known LOM systems based on the DC method. Malasco employs partitioning algorithms and existing matching tools to match large ontologies. It uses three ontology partitioning algorithms: naive algorithm based on RDF sentences, structure-Based algorithm [Stuckenschmidt and Klein2004], and ontology modularity based on $\varepsilon - connection$ [Grau *et al.*2006]. Falcon-AO proposes a structure-based partitioning algorithm to divide ontology elements into a set of small clusters, then constructs blocks by assigning RDF sentences to clusters. We notice that the structure-based partitioning algorithm in Falcon-AO can flexibly control the sizes of modules.

However, DC methods suffer from the contradiction between semantic completeness and information loss. More specifically, after partitioning, ontology elements near boundaries of modules are possible to lose useful semantic information. The more modules we have, the more information will be lost. This may degrade the quality of ontology matching. In Malasco, Paulheim realizes this problem and hence uses overlapping partitions to compensate such information loss. Paulheim claims that the overlapping partitions could limit the loss of precision less than 20% [Paulheim2008]. However, he also points out that overlapping partitions cause the matching phase running up to four times as long as non-overlapping partitions.

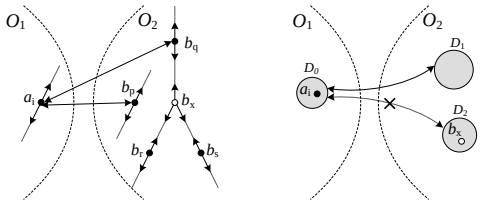
3 Reduction Anchors

During matching large ontologies, we have two interesting observations: (1) A large ontology is often composed of concept hierarchies organized by *is-a* or *part-of* properties, and a correct alignment should not be inconsistent with such hierarchies; (2) An alignment between two large ontologies has locality, i.e., most elements of region D_i in ontology O_1 will match the elements of region D_j in ontology O_2. The two observations provide the new perspective for finding the efficient LOM solution.

In Fig. 1(a), according to the first observation, if a_i matches b_p or b_q, it will have a direct benefit: the subsequent similarity calculations between sub-concepts(/super-concepts) of a_i and super-concepts(/sub-concepts) of b_p or b_q can be skipped. In this paper, we call such concepts like b_p or b_q the positive reduction anchors about a_i, which employ the ontology hierarchy feature to reduce the time complexity in LOM. The positive reduction anchor is defined as follows.

Definition 1 (Positive Reduction Anchor (P-Anchor)) *Given a concept a_i in ontology O_1, let the similarities between a_i and concepts $b_1, b_2, ..., b_n$ in ontology O_2 are $S_{i1}, S_{i2}, ..., S_{in}$, respectively. If S_{ij} is larger than the predefined threshold $ptValue$, the concept pair (a_i, b_j) is a positive reduction anchor, and all positive reduction anchors about a_i are denoted by $PA(a_i) = \{b_j | S_{ij} > ptValue\}$.*

It is clear that positive reduction anchors are symmetrical, i.e. if $b_p \in PA(a_i)$, then $a_i \in PA(b_p)$. $ptValue$ is a larger value in $[0, 1]$.



(a) Positive Reduction Anchor (b) Negative Reduction Anchor

Figure 1: Reduction anchors in large ontology matching

Fig. 1(b) shows the locality phenomenon in LOM, where D_i represents a region in the ontology. Most elements in D_0 are matched to the elements in D_1. Suppose a_i in D_0 does not match b_x in D_2. According to the second observation, we can infer that the neighbors of a_i do not match b_x too. As a result, we can skip the subsequent similarity calculations between the neighbors of a_i and b_x, which will also reduce the times of similarity calculations. In this paper, we call such concepts like b_x the negative reduction anchors about a_i, which employ locality of matching to reduce the time complexity. The negative reduction anchor is defined as follows.

Definition 2 (Negative Reduction Anchor (N-Anchor)) *Given a concept a_i in ontology O_1, let the similarity values between a_i and concepts $b_1, b_2, ..., b_n$ in ontology O_2 are $S_{i1}, S_{i2}, ..., S_{in}$, respectively. If S_{ij} is smaller than the predefined threshold $ntValue$, the concept pair (a_i, b_j) is a negative reduction anchor, and all negative reduction anchors about a_i are denoted by $NA(a_i) = \{b_j | S_{ij} < ntValue\}$.*

It is clear that negative reduction anchors are also symmetrical. $ntValue$ is usually a small value in $[0, 1]$.

Based on positive and negative anchors, ontology mathcing process can skip many similairity calculations, which will significantly reduce the time complexity. Since P-Anchors and N-Anchors cannot be identified in advance, we hence dynamically generate them during ontology matching.

4 Large Ontology Matching Algorithms

4.1 LOM-P: Large Ontology Matching Algorithm Based on P-Anchors

Let $PS(a_i)$, the positive reduction set of a_i, be all the ignorable similarity calculations predicted by $PA(a_i)$. If

$|PA(a_i)| > 0$, we select the *top-k* P-Anchors with maximum similarities. Let $PS(a_i|b_j)$ be the positive reduction set about a P-Anchor (a_i, b_j).

If $PA(a_i) = \{b_p\}$, then $PS(a_i) = [sub(a_i) \otimes sup(b_p)] \cup [sup(a_i) \otimes sub(b_p)]$. Here, $sup()$ and $sub()$ represent the super-concepts and sub-concepts respectively, and $\otimes$ denotes the Cartesian product.

If $PA(a_i) = \{b_q, b_r\}$, then $PS(a_i|b_r) = [sub(a_i) \otimes sup(b_r)] \cup [sup(a_i) \otimes sub(b_r)]$. Let $mid(b_q, b_r)$ be the middle concepts on the hierarchy path from b_q to b_r. Since $sup(b_r) = mid(b_r, b_q) \cup sup(b_q)$, the above formula can be rewritten as:

$$\begin{aligned} PS(a_i|b_r) = \; & [sub(a_i) \otimes sup(b_q)] \cup [sup(a_i) \otimes sub(b_r)] \\ & \cup [sub(a_i) \otimes mid(b_r, b_q)] \end{aligned}$$

Similarly, we obtain:

$$\begin{aligned} PS(a_i|b_q) = \; & [sub(a_i) \otimes sup(b_q)] \cup [sup(a_i) \otimes sub(b_r)] \\ & \cup [sup(a_i) \otimes mid(b_r, b_q)] \end{aligned}$$

Therefore,

$$\begin{aligned} PS(a_i) &= PS(a_i|b_r) \cap PS(a_i|b_q) \\ &= [sub(a_i) \otimes sup(b_q)] \cup [sup(a_i) \otimes sub(b_r)] \end{aligned}$$

Let $lub(b_r, b_q)$ and $glb(b_r, b_q)$ be the least upper bound and the greatest lower bound for b_r and b_q, respectively. The above formula can be simplified as follow:

$$\begin{aligned} PS(a_i) = \; & [sub(a_i) \otimes sup(lub(b_r, b_q))] \\ & \cup [sup(a_i) \otimes sub(glb(b_r, b_q))] \end{aligned}$$

The above analyses can be extended to the general case: given $PA(a_i) = \{b_1, b_2, ..., b_k\}$, the corresponding reduction set can be calculated by:

$$\begin{aligned} PS(a_i) &= \bigcap_{j=1}^{k} PS(a_i|b_j) \\ &= [sub(a_i) \otimes sup(lub(b_1, ..., b_k))] \\ &\quad \cup [sup(a_i) \otimes sub(glb(b_1, ..., b_k))] \end{aligned} \quad (1)$$

Formula (1) indicates that smaller top-k will generate larger $PS(a_i)$. In our implementation, top-k is assigned a value from 1 to 4.

The total positive reduction set during matching is:

$$PS = \bigcup_{i=1}^{n} PS(a_i) \quad (2)$$

The positive reduction set is generated dynamically and consists of two parts: (1) *Invalid positive reduction set* contains all similarity calculations have been computed. Therefore, it is useless for matching; (2) *Valid positive reduction set* contains all similarity calculations to be computed but can be skipped in matching. Therefore, only valid positive reduction set can improve the performance.

The order of similarity calculations will affect the size of the valid positive reduction set. The ideal order can be determined by following theorem.

Theorem 1 *When the order of similarity calculations can divide the hierarchy path L into parts with equal length continually, the P-Anchors can generate the maximum valid positive reduction set with $|L| * (|L| - 2)$ size.*

The proof is omitted for the limitation of space. According to theorem 1, when a path with $|L|$ length generates the maximum positive reduction set, one of the order of similarity calculations is $\frac{L}{2}, \frac{L}{4}, \frac{3L}{4}, \frac{L}{8}, \frac{3L}{8}, \frac{5L}{8}, \frac{7L}{8}, ...$, and it will divide the path into equal lengths continually: $\frac{|L|}{2}, \frac{|L|}{4}, \frac{|L|}{8}, ...$

Algorithm 1 is the large ontology matching algorithm based on P-Anchors (LOM-P). Here, LOMP-Algorithm() is the main function, ComputerSim() matches elements on the hierarchy path recursively, and GetPAnchors() obtains top-k P-Anchors.

Algorithm 1: LOM-P algorithm

Input: ontology O_1, ontology O_2
Output: matching results

```
1  Function LOMP_Algorithm(O1, O2)
2  begin
3      foreach L_i ∈ O1 do
4          ComputeSim(L_i)
5      end
6  end
7  Function ComputeSim(L = (a1, a2, ..., an))
8  begin
9      PA ← GetPAnchors(n/2)
10     PS ← PredictNewPS(PA)
11     ComputeSim(La = (a1, ..., a_(n/2 − 1)))
12     ComputeSim(Lb = (a_(n/2 + 1), ..., an))
13     if |L| ≤ 1 then
14         return
15     end
16 end
17 Function GetPAnchors(a_i)
18 begin
19     foreach b_j ∈ O2 do
20         if (a_i, b_j) ∈ PS then
21             continue
22         end
23         Sim(a_i, b_j) ← Compute(a_i, b_j)
24         if Sim(a_i, b_j) > ptValue then
25             PACandi ← PACandi ∪ b_j
26         end
27     end
28     PA ← MaxTopk(PACandi)
29 end
```

The time complexity of LOM-P algorithm is analyzed as follows. Given two matched ontologies, if all concepts are on a hierarchy path, the matching process can generate $n(n-2)$ size valid positive reduction set, and it just needs $2n$ similarity calculations, i.e., the algorithm has the best time complexity $O(2n)$. However, such ideal case almost does not exist in real world. Suppose there are m hierarchy paths, then the average depth of the ontology is $\bar{d} = \frac{n}{m}$. Consequently, we can derive the time complexity of Algorithm 1 is $O((1 - \frac{1}{m})n^2) = O((1 - \frac{\bar{d}}{n})n^2)$. It means that LOM-P algorithm can improve the matching performance when the ontologies have large average depths.

4.2 LOM-N: Large Ontology Matching Algorithm Based on N-Anchors

N-Anchors are also able to predict ignorable similarity calculations. The set of all ignorable similarity calculations predicted by N-Anchors are called the negative reduction set. Let $Nb(a_i) = \{a_x | d(a_x, a_i) <= nScale\}$ be the neighbors with $nScale$ distance to a_i. Therefore, the negative reduction set generated by a_i is:

$$NS(a_i) = Nb(a_i) \otimes NA(a_i) \quad (3)$$

According to formula (3), $NA(a_i)$ will be propagated to neighbors of a_i. Such propagation will lead to low credible negative reduction set because it will improve the risk of missing right similarity calculations. Fig. 2. interprets the potential risk. Let $NA(a_i) = N_s + N_p$ and $NA(a_j) = N_p + N_q$. If we first calculate the similarities about a_i, then a_j will get $NA(a_i)$. Therefore, we will skip the similarity calculations between a_j and N_s, which would miss correct alignments like $(a_j, b_x)(b_x \in N_s)$. Therefore, N_s is a potential risk for similarity calculations about a_j.

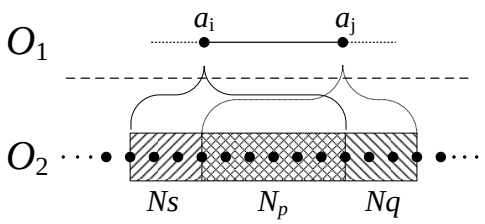


Figure 2: The risk of propagation of N-Anchors

To reduce such risk, we introduce three constraints on the negative reduction set.

Constraint 1: *All N-Anchors must be obtained in similarity calculating*. This means that N-Anchors propagated from neighbors cannot be propagated again. In Fig. 2, $NA(a_j)$ consists of two parts: N_p is propagated from a_i and N_q is obtained in calculating similarity about a_j. Only N_q can be propagated to other neighbors.

Constraint 2: *All N-Anchors of a_i can only be propagated to the neighbors in the semantic subgraph of a_i*. We call this constraint the SSG constraint. Constraint 2 further restricts that N-Anchors can only be propagated to neighbors with close semantic relations to a_i.

Constraint 3: *All N-Anchors of a_i can be propagated only if the description document of a_i contains more than t items.* we call this constraint the SDD constraint. It means that if an element lacks of enough literal information in matching, its N-Anchors may be incorrect and cannot be propagated. In the implementation, we set $t = 8$.

Algorithm 2: LOM-N algorithm

```
   Input: ontology O1, ontology O2
   Output: matching results
1  begin
2      SortConceptByDegree()
3      foreach C_i ∈ O_1 do
4          NA(C_i) ← ∅
5          foreach D_j ∈ O_2 do
6              if (C_i, D_j) ∈ NS then
7                  continue
8              s←ComputeSim(C_i, D_j)
9              if s < ntValue AND InConstraint()
               then
10                 NA(C_i) ← D_j
11             end
12             NS(C_i)← BuiltNS(NA(C_i))
13             NS(C_i)← RefineNS(NS(C_i))
14             NS←NS ∪ NS(C_i)
15         end
16     end
17 end
18 end
```

The order of similarity calculations would also influence the size of negative reduction set. In order to generate the maximum valid negative reduction set, the elements with more neighbors should be calculated first during matching.

Algorithm 2 is the large ontology matching algorithm based on P-Anchors (LOM-P). All concepts are sorted by their degrees (line 2). If a similarity s is smaller than $ntValue$ and satisfies three constraints (line 9), a N-Anchor (line 10) is used to get the negative reduction set (line 12). The valid negative reduction set is obtained after refined (line 13). The time complexity of the algorithm is $O((1 - w\lambda)n^2)$, where w is the average degree and λ is determined by $ntValue$ and constraints. The bigger w and λ, the higher performance the algorithm has.

4.3 LOM-Hybrid: Hybrid Large Ontology Matching Algorithm

We use a hybrid algorithm, called LOM-Hybrid, to combine the LOM-P and LOM-N algorithms to obtain as large valid reduction set as possible. The LOM-Hybrid algorithm will prefer the matching order of LOM-N algorithm for two reasons: (1) Since the average depth of a real ontology is often small, LOM-P may not have the ideal high performance. (2) LOM-N algorithm first calculates the elements with large degree, it can be benefit for the LOM-P algorithm. Therefore, the LOM-Hybrid algorithm is mainly based on the framework of the LOM-N algorithm, in which the LOM-P algorithm is embedded. LOM-Hybrid can generate the valid positive reduction set and valid negative reduction set. Theoretically, the time complexity of LOM-Hybrid is between the complexity of LOM-N and the complexity of LOM-P. Indeed, it is very close to LOM-N. The LOM-Hybrid algorithm is omitted here for the space limitation.

5 Experimental Evaluation

5.1 Data sets

Two data sets are used in the experiments:

- **Dataset1** consists of five middle scale ontology pairs ($russia12$, $russiaAB$, $russiaCD$, $tourismAB$, and $sport$), which are selected from FOAM [2]. These ontologies have 103-474 concepts and 22-100 properties, and will be used to examine the correctness of our algorithms.
- **Dataset2** is the real large scale ontology set in OAEI2008, i.e. $Anatomy$, Fao, and $Library$. These ontologies have 2000-30000 concepts, and will be used to examine the performance of our algorithms.

5.2 Experiments for examining the algorithms

Besides Precision, Recall, and F1-Measure, we also introduce the $Loss$ to measure the quality loss in LOM algorithms:

$$Loss = \frac{F_{gold} - F_{get}}{F_{gold}} \times 100\%$$

where F_{gold} is the F1-measure of a general ontology matching (GOM) algorithm provided by the matching system Lily [3], and F_{get} is the F1-measure of our LOM algorithms.

As shown in Table 1, we have following observations: (1) Three LOM algorithms perform well on middle ontologies.

[2] http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/ontologies.htm
[3] http://cse.seu.edu.cn/people/pwang/lily.htm

Table 1: LOM algorithm VS GOM algorithm (on FOAM data set)

	GOM	LOM-P		LOM-N		LOM-Hybrid	
	F1-Measure	F1-Measure	Loss	F1-Measure	Loss	F1-Measure	Loss
Russia12	0.72	0.71	1%	0.67	7%	0.68	6%
RussiaAB	0.99	0.99	0%	0.96	3%	0.97	2%
RussiaCD	0.94	0.91	3%	0.85	10%	0.85	10%
TourismAB	0.89	0.88	1%	0.85	4%	0.85	4%
Sports	0.75	0.77	-3%	0.75	0%	0.75	0%
Average	0.86	0.85	0.4%	0.82	4.8%	0.82	4.4%

Table 2: Matching results on Anatomy

System	Runtime	Precision	Recall	F1-Measure	Recall+
Label Eq.	–	0.98	0.61	0.76	0.00
LOM-Hybrid	13min	0.80	0.70	0.75	0.47
Falcon-AO	12min	0.96	0.60	0.74	0.13
TaxoMap	25min	0.46	0.76	0.57	0.47

(2) LOM-P, LOM-N and LOM-Hybrid have similar performance, which can be generally sorted by: LOM-P≥LOM-Hybrid≥LOM-N. (3) Compared with GOM algorithm, three LOM algorithms have a certain quality loss. The average $Loss$ of LOM-P, LOM-N and LOM-Hybrid are 0.4%, 4.8% and 4.4%, respectively. We set LOM-Hybrid as the standard LOM algorithm in remaining experiments, because it not only combines other two algorithms, but also has a close running time to LOM-N.

We also compare the LOM-Hybrid with other matching system using DC method. We select Falcon-AO as the comparing system, whose matching method called PBM. As shown in Fig. 3, LOM-Hybrid has a similar performance to PBM.

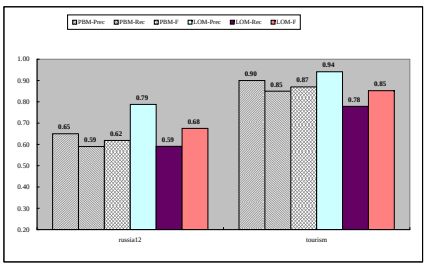


Figure 3: LOM-Hybrid VS Falcon-AO-PBM

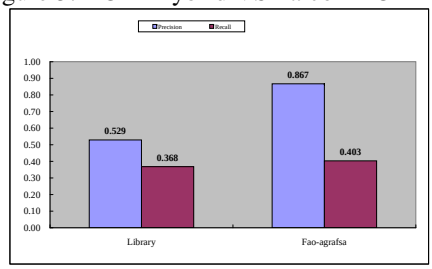


Figure 4: Matching results on the real large ontologies

5.3 Real large ontology matching experiments

Here we present results of our LOM algorithms on three LOM tasks ($Anatomy$, Fao, and $Library$) in OAEI2008.

$Anatomy$ task requires matching two ontologies with 2700 and 3300 concepts. There are 13 systems that participate in the anatomy task, but only few systems use the special large ontology matching method. The TaxoMap uses the PBM algorithm of Falcon-AO. We employ the $Recall+$ [Caracciolo *et al.*2008] to measure how many non trivial correct alignments can be found. From Table 2, we can see that: (1) LOM-Hybrid can also perform well in $Anatomy$ task. Indeed, it outperforms than some systems using background knowledge. (2) LOM-Hybrid and Falcon-AO have similar performance, which are better than TaxoMap. (3) The running time of LOM-Hybrid is 13 minutes. It indicates that our LOM algorithms have similar running time to other systems using DC methods. (4) Our method and Taxomap have a high Recall+, indicating that they have the ability to discover the difficult alignments.

$Library$ task contains two ontologies with 5000 and 35000 concepts. Fao contains several ontologies with 2000 to 10000 concepts. As Fig. 4 shows, our method can discover the alignments for the two tasks.

5.4 Performance experiments

There is a need to analyze the influence of key parameters to our algorithms. Here we use a new metric G called *benefit rate* to measure how much a LOM algorithm can improve the performance: $G = \frac{N}{n_1 \times n_2}$, where N is the size of total reduction set, n_1 and n_2 represent the number of concepts in two ontologies. The larger G is, the fewer times of similarity calculations and the higher efficiency of the algorithm has. Although we use Dataset1 to perform the following experiments, it should be noted that similar conclusions can be obtained from other datasets.

top-k is a key parameter in LOM-P. We need to know the influence of different *top-k* values to the quality of results and efficiency. Our experimental results show that there is a small variation of matching results when *top-k* changes from 1 to 4. This result can be explained that most elements only have very few P-Anchors. Since ontologies in Dataset1 have short hierarchy pathes, LOM-P just gets about 5% benefit rate.

LOM-N algorithm has four important parameters: $ntValue$, $nScale$, SDD constraint and SSG constraint. We evaluate these parameters on the $TourismAB$ of Dataset1.

Fig. 5 shows the relation of $ntValue$ to F1-Measure on different $nScale$. Fig. 6 shows the relation of benefit rate to $ntValue$ under different $nScale$. We observe that: (1) $ntValue$ has a significant effect on matching quality and efficiency, i.e., larger $ntValue$ will lead to lower matching qual-

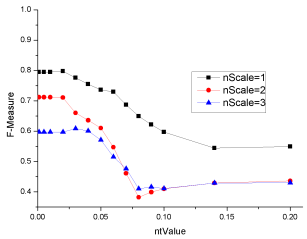


Figure 5: $ntValue$ - $nScale$ - matching quality

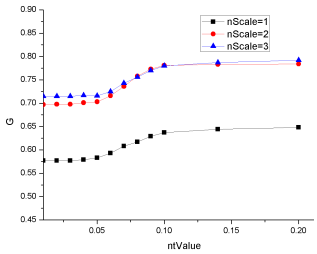


Figure 6: $ntValue$ - $nScale$ - benefit rate

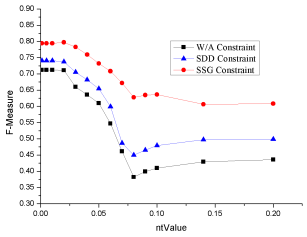


Figure 7: SDD - SSG - matching quality

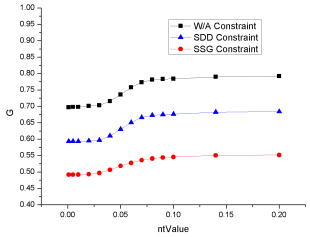


Figure 8: SDD - SSG - benefit rate

ity, meanwhile, it also causes higher benefit rate. (2) $nScale$ also affects the matching quality and efficiency. As $nScale$ increases, matching quality will decrease, but the benefit rate will increase. Results also show that $ntValue < 0.02$ and $nScale = 2$ will lead to a good matching quality and benefit rate. Fig. 7 and Fig. 8 show the influences of SDD and SSG on matching quality and benefit rate. The line with rectangle represents the results without any constraint. We can see that: (1) Under three constraints, the matching quality will increase, but the benefit rate decreases. (2) SSG constraint has a higher influence on matching quality and benefit rate.

6 Conclusion

This paper proposes a new efficient large ontology matching method based on reduction anchors. Reduction anchors are used to predict the ignorable similarity calculations in matching. Our experimental results show that the proposed method is effective for matching large ontologies.

Acknowledgments

The work is supported by the NSF of China (61003156 and 90818027), the National High Technology Research and Development of China (863 Program) (2009AA01Z147), and the Major State Basic Research Development Program of China (973 Program) (2009CB320703)

References

[Caracciolo *et al.*, 2008] Caterina Caracciolo, Jrme Euzenat, Laura Hollink, Ryutaro Ichise, and et al. Results of the ontology alignment evaluation initiative 2008. In *The Third International Workshop on Ontology Matching (OM2008), Karlsruhe, Germany.*, 2008.

[Grau *et al.*, 2006] B. Cuenca Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Modularity and web ontologies. In *Proc. KR-2006*, 2006.

[Grau *et al.*, 2007] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Just the right amount: extracting modules from ontologies. In *Proceedings of the 16th international conference on World Wide Web (WWW2007)*, 2007.

[Hovy, 1998] Eduard Hovy. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC98)*, 1998.

[Hu and Qu, 2006] Wei Hu and Yuzhong Qu. Block matching for ontologies. In *The 5th International Semantic Web Conference (ISWC2006)*, 2006.

[Hu *et al.*, 2008] Wei Hu, Yuzhong Qu, and Gong Cheng. Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering*, 67(1):140–160, 2008.

[Mao, 2008] Ming Mao. *Ontology Mapping: Towards Semantic Interoperability in Distributed and Heterogeneous Environments*. PhD thesis, University of Pittsburgh, 2008.

[Massmann and Rahm, 2008] Sabine Massmann and Erhard Rahm. Evaluating instance-based matching of web directories. In *The 11th International Workshop on Web and Databases 2008 (WebDB2008)*, 2008.

[Mork and Bernstein, 2004] Peter Mork and Philip A. Bernstein. Adapting a generic match algorithm to align ontologies of human anatomy. In *Proceedings of the 20th International Conference on Data Engineering (ICDE2004)*, 2004.

[Paulheim, 2008] Heiko Paulheim. On applying matching tools to large-scale ontologies. In *The Third International Workshop on Ontology Matching*, 2008.

[Stuckenschmidt and Klein, 2004] Heiner Stuckenschmidt and Michel Klein. Structure-based partitioning of large concept hierarchies. In *Third International Semantic Web Conference (ISWC2004)*, 2004.

[Zhang *et al.*, 2007] Songmao Zhang, Peter Mork, Olivier Bodenreider, and Philip A. Bernstein. Comparing two approaches for aligning representations of anatomy. *Artificial Intelligence in Medicine*, 39:227–236, 2007.