Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 8107968, 10 pages
https://doi.org/10.1155/2020/8107968

Research Article

Matching Large Scale Ontologies Based on Filter and Verification

Yingxin Li,[1] Zhou Jianhui,[2] Jihong Liu,[3] and Yongzhu Hou[3]

[1]*School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China*
[2]*School of Computer and Network Engineering, Shanxi Datong University, Datong 037009, China*
[3]*School of Mechanical Engineering and Automation, Beihang University, Beijing 100191, China*

Correspondence should be addressed to Zhou Jianhui; zhjh19851101@163.com

Received 9 December 2019; Revised 20 March 2020; Accepted 16 April 2020; Published 22 May 2020

Academic Editor: Gaetano Giunta

Ontology matching is an effective method to realize intercommunication and interoperability between heterogeneous systems. The essence of ontology matching is to discover the similar entity pairs between source ontology and target ontology, which is a process calculating the similarity between entities in ontologies. The similarity can be calculated utilizing various features between entity pairs, such as string similarity, structural similarity, and semantic similarity. The larger the ontology scale, the lower the efficiency and accuracy rate of ontology matching. As the ontology scale increases, the amount of entities in ontologies will be larger and the ontologies will become more heterogeneous. This paper proposes an innovative method of matching large scale ontologies based on filter and verification, which firstly reduces the heterogeneous of large scale ontologies in the filter phase and then matches the reduced ontologies in the verification phase. Large scale ontologies will be partitioned into several subontologies to get a proper scale before matching. The benchmark of Anatomy and Food in OAEI is adopted to evaluate the proposed method, and the experimental result illuminates that the recall rate is improved in the situation of retaining efficiency and accuracy rate using the proposed method.

1. Introduction

Ontology matching can solve the problems of intercommunication and interoperability between heterogeneous systems. With the development of the Internet and information technologies, the volume of data is growing rapidly, which leads to the increase of the scale of ontology. Because the scale of ontology is becoming larger, the complexity of the ontology matching algorithm increases exponentially; as a result, the efficiency of ontology matching becomes lower. In addition, the increase of the ontology scale enhances the degree of heterogeneity between ontologies. The greater the degree of heterogeneity, the more difficult it is to discover similar entities between ontologies, as the neighbouring entities have more heterogeneous information. It also results in that accuracy rate and recall rate of ontology matching is lower.

The idea of partitioning ontology into several subontologies before matching them can effectively reduce the scale of ontology beforehand and reduce the degree of heterogeneity between ontologies, which can not only improve the efficiency of ontology matching but also improve the accuracy rate and recall rate of ontology matching. For instance, between source ontology and target ontology, a large number of dissimilar entities exist, which are named as irrelevant entities pairs. It is the core idea of our proposed method that how to recognize the irrelevant entities pairs without calculating their similarities. As shown in Figure 1, the entity named *concentration* in source ontology is irrelevant to the entity named *trunk* in target ontology, which is generally found by calculating the integrated similarity between the entities named *concentration* and *trunk*. However, this result can be found according to the fact that neighbouring entities of *concentration* and neighbouring entities of *trunk* are irrelevant, so it is unnecessary to calculate integrated similarities. Removing the irrelevant entity pairs from matching candidates will reduce the heterogeneous degree between source ontology and target ontology. In this paper, a method of matching large-scale ontologies based on filter and verification is proposed, in which both

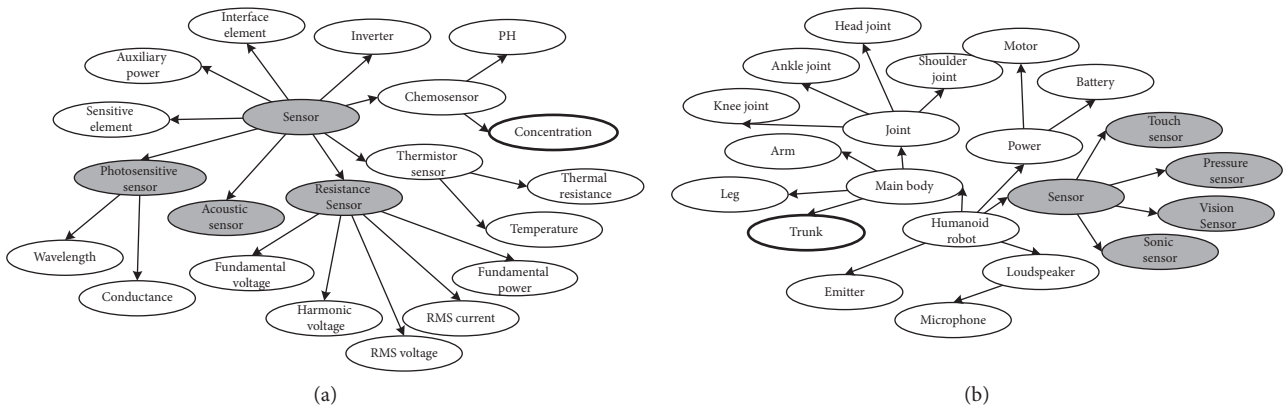

Figure 1: Instance of heterogeneous ontologies. (a) Source ontology. (b) Target ontology.

source ontology and target ontology are partitioned into several subontologies firstly, then these subontologies are matched, and finally the alignments of every subontology matching are integrated.

Although reducing the scale of ontologies before matching them can improve the efficiency of ontology matching, it will lead to the lower recall rate of ontology matching because some pairs of entities are previously considered irrelevant, but they are actually related. How to improve the recall rate in the situation of retaining the efficiency is a key problem in large-scale ontology matching. The main idea of solving this problem is that the lower irrelevant probability, the more chance is given to calculate their similarities. In other words, entities at the margin of one subontology should be allowed to appear in other subontologies so that chances of calculating the similarities of these entities will increase. Compared with the entities of large-scale ontology, the number of these entities is small, so the efficiency is retained reasonably in this paper.

The remainder of the paper includes five sections. Section 2 introduces some related works about large-scale ontologies matching. Section 3 introduces the overall framework of the proposed method. In Section 4, each step in the filter phase for partitioning ontologies is elaborated. The verification phase is illuminated in Section 5, in which subontologies are matched. Finally, in Section 6, the proposed method is evaluated by the benchmark of Anatomy and Food.

2. Related Works

The process of ontology matching is to calculate the similarity between entities in source ontology and target ontology. The similarities include string similarity, structural similarity, and semantic similarity [1]. String similarity was calculated by the distance between label strings of entities, such as edit distance, Euclidean distance, and Hamming distance [2]. Structural similarity was calculated by utilizing the defined hierarchies and concept connections between entities [3]. Semantic similarity was calculated by depending on the external resources, such as WordNet and other external ontologies [4].

Three types of similarities exist simultaneously between entities in source ontology and target ontology. There are two approaches to match ontologies. The first approach was to generate part of alignments by calculating string similarities between entities, and then to generate the remaining alignments by calculating structural similarities or sematic similarities between entities [5, 6]. The second approach was to integrate the three type similarities between entities. In [7], an adaptive method was deployed to obtain the weights of three type similarities, and an artificial neural network was utilized to integrate them.

According to the conclusion of OAEI report [8], in reference alignments, about 30% alignments were obtained by calculating the string similarities between entities in source ontology and target ontology, about 50% alignments were obtained by calculating the structural similarities between entities, and about 20% alignments were obtained by calculating the semantic similarities between entities. Therefore, it is important to calculate structural similarities.

With the growth of information, the scale of ontology is increasing rapidly. The number of entities and relationships between entities are also increasing. For instance, the number of entities in Anatomy and Food was more than 2500 and 20000, respectively [9]. In addition, the scale of some domain ontologies in complex product design fields was large, such as aircraft, robots, and vehicle domain ontologies. The number of entities in them was usually 7000–8000, and the relationships between entities were more than 30000 [10].

The larger the scale of ontology, the greater the heterogeneity between ontologies and the more difficult to utilize structural information to calculate structural similarities between entities [11]. As mentioned before, 50% alignments were obtained by calculating structural similarities between entities. Thus, the recall rate of large-scale ontology became lower because of the high heterogeneity between ontologies. At present, the recall rate of large-scale ontology matching reaches only 40%–50% [12].

In the research of large-scale ontology matching, there are two kinds of approaches. The first is based on distributed computing, which parallelized the computing process to improve the computing efficiency [13]. The second is to

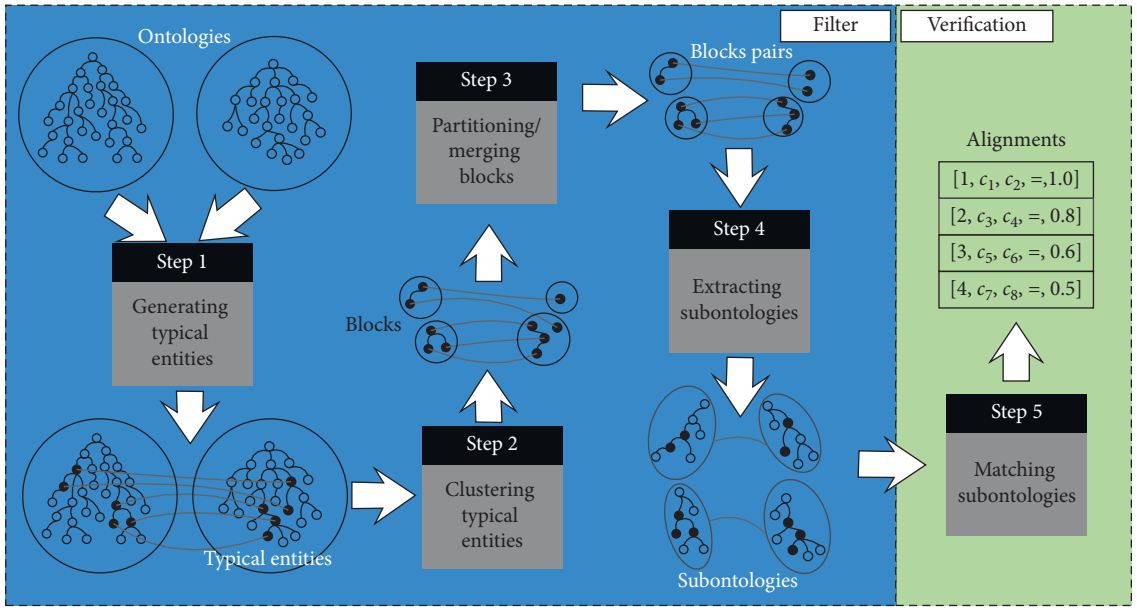

Figure 2: The framework of the proposed method.

partition large-scale ontology into several blocks before matching them. It reduced the computational complexity at the algorithm level [14, 15]. Research in this paper belongs to the second kind of approaches.

In [14], a machine learning strategy for generating possible candidate pairs was designed to filter the uncorrelated entities pairs, and then logical consistency checking was deployed to generate final alignments. The main idea of the machine learning strategy was that pairs of entities were possible candidates if their neighbouring pair of entities has high string similarity. While the degree of heterogeneity between ontologies is increasing, as previously mentioned, a mass of entities lost their candidate access by this strategy, which results in reducing the recall rate of ontology matching. In [15], a divide-and-conquer approach was proposed to match large-scale ontologies. This approach firstly partitioned large-scale ontology to some blocks and then matched pairs of blocks to generate alignments. However, the blocks were not aligned; some correct alignments lost matching access because they were in different pairs of blocks, which was the main reason of reducing the recall rate of ontology matching. According to the above, the proposed method in this paper aligns the blocks by partitioning or merging blocks before matching them and extends the blocks to subontologies. Entities are allowed to appear repetitively in different subontologies. Although it is a little bit time consuming, the recall rate of ontology matching improves obviously.

In this paper, a method of matching large-scale ontologies based on filter and verification is proposed. It reduces the heterogeneity by partitioning ontologies into several subontologies not blocks. Compared with the method that partitioned ontologies into blocks, the proposed method has two differences. On one hand, it is no need to traverse all entity pairs between ontologies, but it has to search relevant entities. On the other hand, the entities are allowed to appear in different subontologies repetitively which is the key cause to improve the recall rate of large-scale ontology matching.

3. Framework

The framework of the proposed method deploys the filter and verification strategy whose main idea is to reduce the scale of ontologies before matching them. As shown in Figure 2, the framework contains two main phases, one is the filter phase and the other is the verification phase.

In the filter phase, the reduction of the scale of ontologies is completed through 4 steps:

(1) Generating typical entities: the inputs are source ontology and target ontology. The outputs are some pairs of entities whose name are typical entities, which have high string similarities.

(2) Clustering typical entities: the inputs are typical entities. The outputs are some blocks of typical entities, which are obtained by the method of clustering.

(3) Partitioning/merging blocks: the inputs are some blocks of typical entities. The outputs are pairs of blocks between ontologies.

(4) Extracting subontologies: the inputs are some pairs of blocks. The outputs are pairs of subontologies, which are obtained by the extension method.

In the verification phase, each pair of subontologies is matched through calculating integrated similarities, which include structure similarities and sematic similarities.

In order to elaborate more details of the framework, a pseudocode description is presented in Algorithm 1. The novelty of the proposed framework is mainly in Step 1, Step 3, and Step 4. In Step 1, length and prefix of entities are utilized to generate typical entities without traversing all pairs of entities between source ontology and target ontology. In Step 3, heterogeneous blocks are transformed to homogeneous blocks by the method of partitioning or

```
Inputs: Source ontology (SO) and Target ontology (TO)
Outputs: Alignments (A)
Variables:ik
hashmap<(length, prefix), ID>//save length and prefix of entities in SO and TO
typicalList < ID_eso, ID_eto>//save typical entities in SO and TO
blockSetSO//save the set of blocks of SO
blockSetTO//save the set of blocks of TO
subOntoSetSO//save the set of sub-ontologies of SO
subOntoSetTO//save the set of sub-ontologies of TO
 (1) //Step 1: generate typical entities
 (2) for entity ei in SO, TO:
 (3)    length = number of letters in label of ei
 (4)    prefix = prefix of label of ei
 (5)    hashmap.put(length, prefix, ID_ei)
 (6) end for
 (7) for key (length, prefix) in hashmap:
 (8)    for IDs in values (ID_e1, ID_e2, . . ., ID_en):
 (9)       if simT(ID_ei, ID_ej) > threshold and ID_ei ∈ SO and ID_ej ∈ TO://simT is shown in formula (2)
(10)          typicalList.add(ID_ei, ID_ej)
(11)       end if
(12) end for
(13) end for
(14) //Step 2: clustering typical entities
(15) for every entities ID_eso, ID_eto in typicalList < ID_eso, ID_eto>:
(16)    setSO.add(ID_eso), setTO.add(ID_eto)
(17)    blockSetSO = partition(setSO), blockSetTO = partition(setTO)//partition is shown in [15].
(18) end for
(19) //Step 3: partitioning/merging blocks
(20) for every entity sets Si in blockSetSO:
(21)    for every entities ID_eso[k] in Si:
(22)       tempEntity = typicalList.get(indexOf(ID_eso[k])).get(1)
(23)       if k = 0:
(24)          //correSet records the corresponding setTO in blockSetTO
(25)          correSet = setTO which contains tempEntity in blockSetTO
(26)       else if tempEntity is not in correSet:
(27)          //sourceSet records the corresponding setSO in blocksSetSO
(28)          sourceSet = setSO which contains tempEntity in blockSetSO
(29)          correSet.add(tempEntity)
(30)          sourceSet.delete(tempEntity)
(31)       end if
(32)    end for
(33) end for
(34) //Step 4: extracting sub-ontologies
(35) for every entity sets Si in blockSetSO, blockSetTO:
(36)    do:
(37)       n = Si.size
(38)       tempSet = null
(39)       //rH, rC, ek is shown as formula (3)~(8)
(40)       for every entities ID_eso[k] in Si:
(41)       candidateEntity = ID_eso[k].get(rdfs:sub-ClassOf)
(42)          if rH(candidateEntity, ID_eso) > threshold:
(43)             tempSet.add(candidateEntity)
(44)       candidateEntity = ID_eso[k].get(rdfs: hasSomeValueFrom)
(45)          if rC(candidateEntity, ID_eso) > threshold:
(46)             tempSet.add(candidateEntity)
(47)       update extension factor ek
(48)       end for
(49)       Si.add(tempSet)
(50)    while(Si.size ! = n)
(51) end for
(52) subOntoSetSO = blockSetSO
```

Algorithm 1: Continued.

```
(53) subOntoSetTO = blockSetTO
(54) //Step 5: matching sub-ontologies
(55) for every pair sub-ontologies (subSO[i], subTO[i]) in subOntoSetSO, subOntoSetTO:
(56)     //matchStruc by V-DOC in [18].
(57)     structureAlignment = matchStruc(subSO[i], subTO[i])
(58)     //matchSema by GMO in [19].
(59)     sematicAlignment = matchSema(subSO[i], subTO[i], structureAlignment)
(60)     A.add(structureAlignment), A.add(sematicAlignment)
(61) end for
```

Algorithm 1: Pseudocode description of the framework.

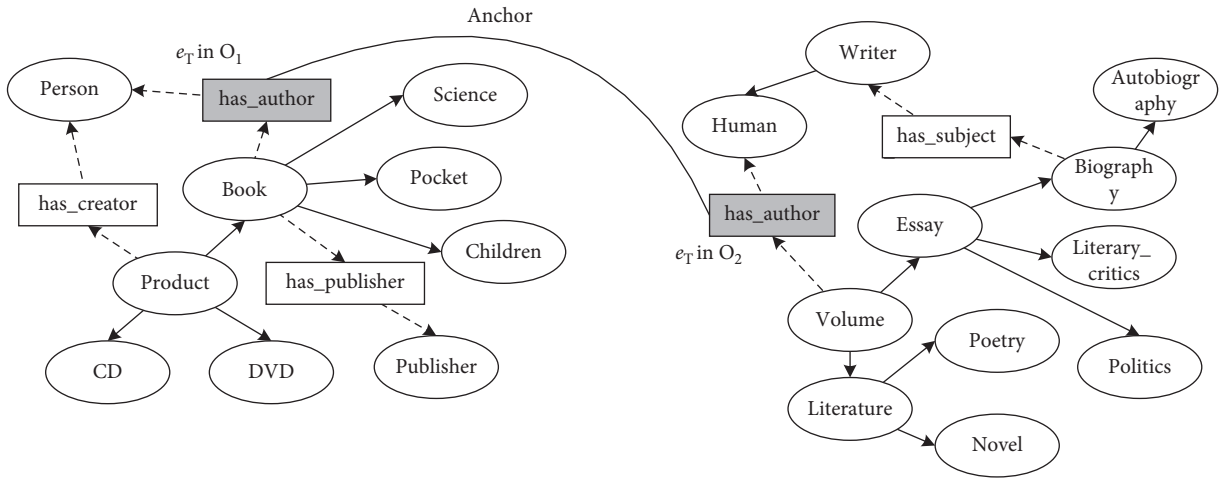

Figure 3: Typical entities' sample.

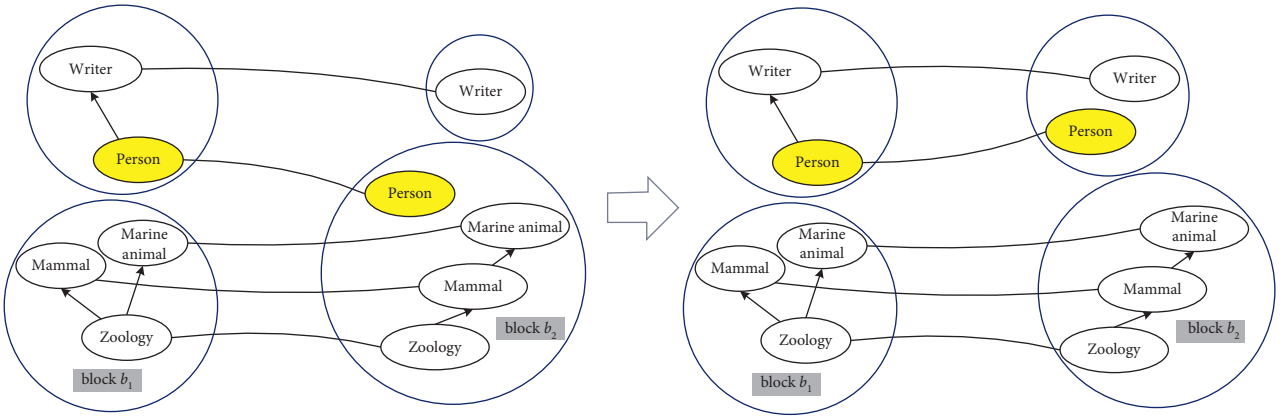

Figure 4: Typical entities' block partition.

merging blocks, ensuring that better subontologies can be extracted in the subsequent step. In Step 4, a method of extracting subontologies is proposed which allows entities at the margin of subontologies to repetitively appear in different subontologies. That improves the recall rate of the final matching step.

According to the pseudocode in Algorithm 1, the proposed approach has two circulations at most. So, its algorithm complexity is O (n^2).

4. Reduction of Large-Scale Ontologies in Filter Phase

Step 1 (generating typical entities).

Alignment is the result of ontology matching, and it is made up of a set of correspondences. Each correspondence is a tetrad, which is denoted as follows:

$$cor = \{e_1, e_2, r, s\}, \tag{1}$$

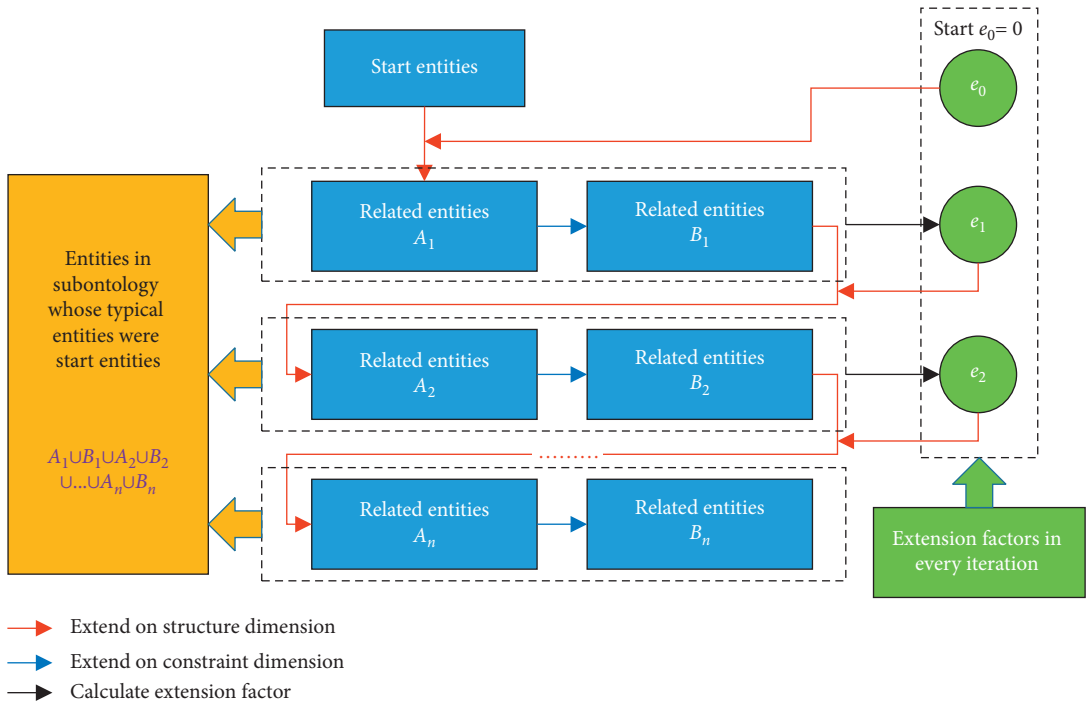

Figure 5: Extension method of extracting subontology.

where *cor* is the correspondence; e_1 and e_2 are entities in source ontology and target ontology, respectively; r is relationship between e_1 and e_2; and s is similarity between e_1 and e_2.

Typical entities (e_T) are referred to the entities in the correspondences whose similarities are high. In [15], the correspondences whose similarity was high were called anchors. Therefore, as shown in Figure 3, e_T are the entities which are connected by anchors, and e_T can be generated by finding anchors. As the ontologies are large, finding anchors by calculating each pair of entities between full ontologies is time consuming. A similarity join algorithm was employed to find anchors, and the framework of this algorithm was a filter-verification framework based on signature, and it used as a light-weight filtering technique to identify a set of candidate pairs and prune lots of dissimilar pairs [16]. In the filter step, length filtering and prefix filtering were both employed to prune the dissimilar pairs; in the verification step, Edit distance was utilized to measure the string similarity between entities. Let r and s be two strings, and the similarity between r and s was denoted as $\mathrm{sim}_T(r, s)$, and it was determined as follows:

$$\mathrm{sim}_T(r,s) = \mathrm{comm}(r,s) - \mathrm{diff}(r,s) + \mathrm{winkler}(r,s), \quad (2)$$

where comm (r, s) and diff (r, s) are the commonality and difference between r and s, respectively, and winkler (r, s) is the correction coefficient introduced in [17].

Step 2 (clustering typical entities).

Through the previous step, some typical entities are obtained in source ontology and target ontology respectively. Employing the ontology partition algorithm in [15], typical entities in the source ontology and target ontology are clustered, respectively, to some typical entities blocks, as shown in Figure 4.

Step 3 (partitioning/merging blocks).

Through the previous step, some blocks in ontologies are obtained, whose entities have anchors between each other. If entity e_i in block b_1 has the anchor to entity e_j in block b_2, then b_1 and b_2 are connected, which denoted as $b_1 \longrightarrow b_2$.

For each e_i in b_1, if only one e_j in b_2 has the anchor to it, and for each e_j in b_2, if only one e_i in b_1 has the anchor to it, then b_1 and b_2 are regarded as homogeneous, else they are inhomogeneous.

If b_1 and b_2 are inhomogeneous, then b_1 or b_2 need to be partitioned with the purpose of changing them to be homogeneous, as shown in Figure 4.

Step 4 (extracting subontologies).

Through the previous step, some pairs of homogeneous blocks are obtained. In this step, taking typical entities in the blocks as input, the extension method is deployed to extract subontologies from source ontology and target ontology, respectively. In other words, if n subontologies are extracted in source ontology, then n subontologies are extracted in target ontology correspondingly.

The extension method of extracting subontology is shown in Figure 5. Taking typical entities in blocks as start entities' set, the extension on structure dimension and constraint dimension are carried out alternately and repetitively. The extension factor is updated in iteration. Iteration is terminated when the extension result no longer changes.

On structure dimension, extension is executed by searching relationship whose label is *rdfs: sub-ClassOf* in ontology. With

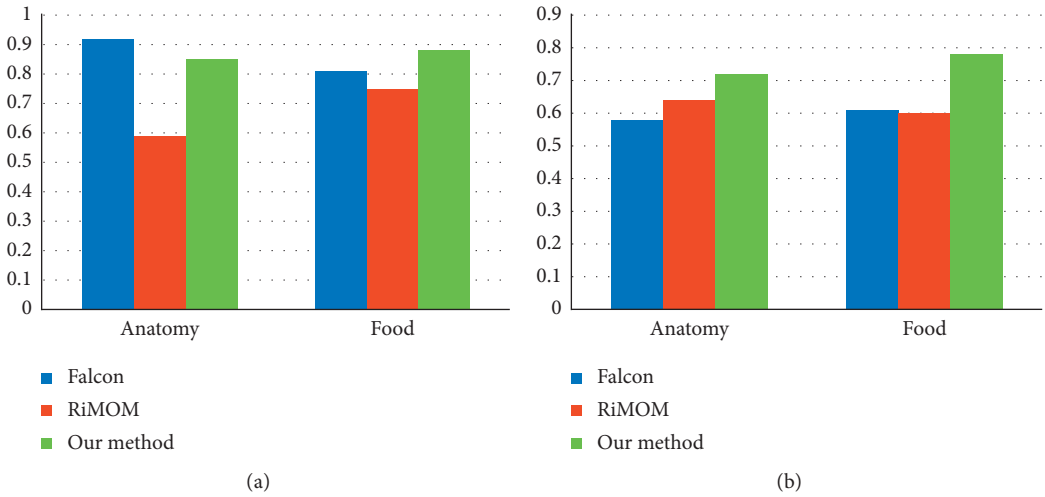

Figure 6: Comparison between falcon, RiMOM, and the proposed method on (a) accuracy and (b) recall.

these relationships, the structure of entities is hierarchy. The similarity between entities which have hierarchical structure can be calculated by (3), where c_i and c_j are entities labels; depth (x) is the depth of entity whose label is x; and c_{ij} is deepest common parent entity of c_i and c_j. On this basis, the relevancy between entities can be calculated by (4), where e is the extension factor, and it will be illustrated in the subsequent content:

$$\mathrm{sim}\left(c_i, c_j\right) = \frac{2\mathrm{depth}\left(c_{ij}\right)}{\left(\mathrm{depth}\left(c_i\right) + \mathrm{depth}\left(c_j\right)\right)}, \tag{3}$$

$$r_{\mathrm{H}}\left(c_i, c_j\right) = \mathrm{sim}\left(c_i, c_j\right) \times (1 + e). \tag{4}$$

If relevancy r_{H} (c_i, c_j) is greater than threshold given artificially, correlative entities are selected as a set, whose name is related entities' set, denoted as A_k, where k represents the kth iteration.

On constraint dimension, taking A_k as start entities' set, extension is executed by searching relationship whose label is *rdfs:hasSomeValueFrom* or *rdfs:hasAllValueFrom* in ontology. With these relationships, entities are correlated by constraints. The relevancy between entities equals the similarity between them, which can be calculated by (5), where c_i' is entity who has constrain to A; depth'(x) is the virtual depth of entity whose label is x; and Ac_i is deepest common parent entity of A and c_i. Among them, virtual depth is the depth of entity whose constraint relationship is regard as hierarchy relationship. For example, if the depth of entity U was 3, and one constraint relationship between entity V and U existed, then V was regard as the parent entity or child entity, and the virtual depth of entity V was 2 or 4. Virtual depth can be calculated by (6):

$$r_{\mathrm{C}}\left(A, c_i\right) = \mathrm{sim}\left(A, c_i\right) = \frac{2\mathrm{depth}\left(Ac_i\right)}{\left(\mathrm{depth}\left(A\right) + \mathrm{depth}'\left(c_i'\right)\right)}, \tag{5}$$

$$\mathrm{depth}'\left(c_i'\right) = \begin{cases} \mathrm{depth}\left(c_i\right) + 1 & c_i \text{ has Value From } c_i', \\ \mathrm{depth}\left(c_i\right) - 1 & c_i' \text{ has Value From } c_i. \end{cases} \tag{6}$$

Table 1: Comparison between falcon, RiMOM, and the proposed method on efficiency.

	Falcon (min)	RiMOM (min)	Our method (min)
Anatomy	7	75	7
Food	23	240	28

Like extension on structure dimension, if relevancy r_{C} (A, c_i) is greater than threshold given artificially, correlative entities are selected as a set, whose name is related entities set, denoted as B_k, where k represents the kth iteration. B_k is the start entities set of the extension on structure dimension in the $k+1$th iteration.

Extension factor refers to the ratio of increase on constraint dimension between B_k and S, and S is the start entities' set. It can be calculated after each iteration by (7), and then in next iteration, the extension factor is applied to calculate relevancy on structure dimension, as shown in (8):

$$e_k = \frac{\left(r_C^k\left(S, B_k\right) - \mathrm{sim}\left(S, B_k\right)\right)}{\mathrm{sim}\left(S, B_k\right)}, \tag{7}$$

$$r_H^{(k+1)}\left(B_k, c_i\right) = \mathrm{sim}\left(B_k, c_i\right) \times \left(1 + e_k\right). \tag{8}$$

In (7) and (8), after the kth iteration, e_k is calculated, and in the $k+1$th iteration, e_k is applied.

While extension iteration is terminated, overall A_k and B_k are union as one entities' set, in which overall entities compose one subontology.

After Step 4, source ontology and target ontology are partitioned to several pairs of subontologies. Subontology has two characteristics: (1) one subontology in source ontology must correspond one subontology in target ontology. It is one-to-one, and it means that only the similarities of entities which come from one-to-one pair of subontologies will be calculated. That retains the efficiency of large-scale ontology matching. (2) Some entities may appear in more than one

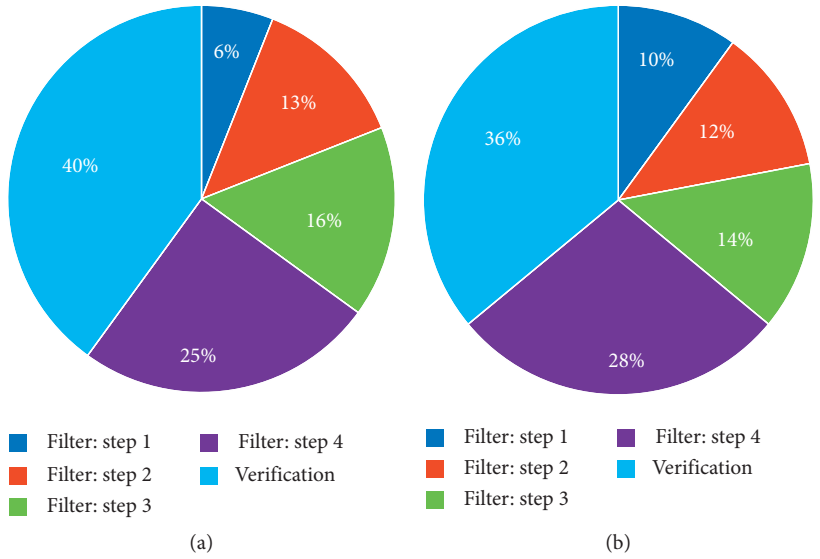

Figure 7: Execution time of every steps in filter and verification phases. (a) Anatomy. (b) Food.

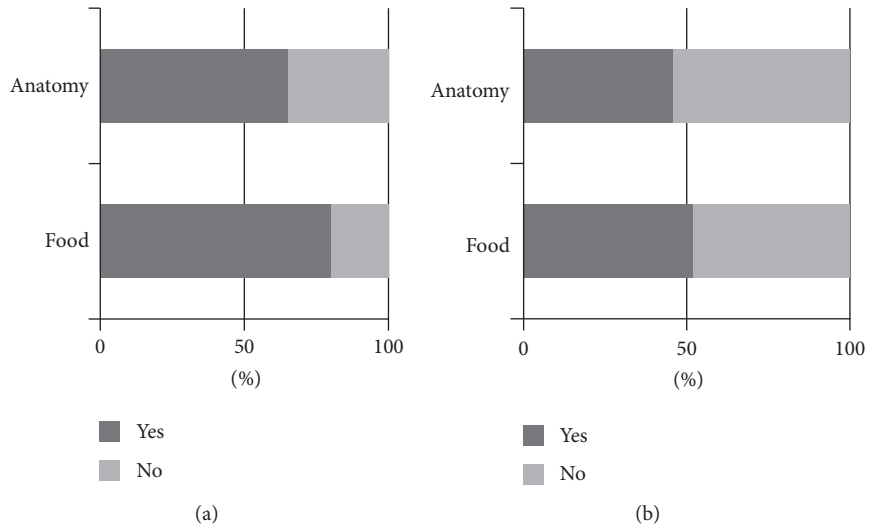

Figure 8: Percent of entities participating in calculation in (a) filter phase and (b) verification phase.

subontologies. It is vague that these entities are or not irrelevant to other entities. Therefore, more subontologies contain them, more chance of calculating they have. That improves the recall rate of large-scale ontology matching.

5. Matching Subontologies in Verification Phase

Step 5 (matching subontologies).

In the previous phase, same number subontologies in source ontology and target ontology are obtained. In this phase, each pair of subontologies between them will be matched, and the result will be union as the alignments between them.

Structure similarity and semantic similarity are calculated between entities in subontologies, respectively. Take each pair of subontologies as input, and V-DOC and GMO are deployed to generate internal alignments and external alignments, respectively. About structure similarity, V-DOC is deployed to generate internal alignments between subontologies, whose novelty lies in the construction of virtual documents [18], and then about semantic similarity, GMO is deployed to generate external alignments between subontologies, which use RDF Bipartite Graphs to represent ontologies [19]. Finally, the internal and external alignments are union as the output.

6. Evaluation and Discussion

In our research, we developed the Dynamic Ontology Matching System (DYOM), in which the core component of

ontology matching used the proposed method in this paper. DYOM is an offline system and it was encoded by Java 1.7. DYOM has been employed in a robot design project mentioned in the Acknowledgments section.

Anatomy and Food are selected as benchmark datasets provided by OAEI organization. Anatomy and Food can be obtained from the website mentioned in the Data Availability section. In Anatomy track, ontologies denoted as *Human* and *Mouse* were input to the algorithm. *Human* contains 3304 entities and *Mouse* contains 2743 entities. Food track includes two ontologies denoted as *AGROVOC* and *NALT*. *AGROVOC* contains 28439 entities and *NALT* contains 42326 entities. The scale of these two datasets are orders of 1000 and 10000, respectively.

Two large ontology matching system named Falcon and RiMOM are adopted to be compared with our method because the performance of these two system is better than others in large-scale ontology matching, according to the conclusion in 2007 OAEI report. The reason of choosing 2007 OAEI report is that there are very few research studies on large-scale ontology matching and performance of these few systems was almost the same with Falcon and RiMOM after 2007 [20]. In addition, the criteria of accuracy rate, recall rate, and efficiency were included in 2007 OAEI report, which can be obtained from the website mentioned in the Data Availability section.

In the proposed method, there is only one parameter, which is the threshold given artificially in our system. We took 0.75 as the value of threshold because Falcon took 0.75 also. It ensured that the evaluation about accuracy rate and recall rate is a baseline test. As shown in Figure 6, our method improves the recall rate of large-scale ontology matching, and the accuracy rate is almost same as the other systems.

In order to test time consumption between our method and others, we try to adopt the computer configuration at 2007 (CPU Intel Core 2 Duo E4300, 1 GB DDR2, Inno3D 7900GS). However, it is not a baseline test that is on the efficiency between our method and Falcon, RiMOM. The results of this experiment are only for reference. Considering about order of magnitude, as shown in Table 1, our method has almost the same efficiency with the other methods.

We also test the execution time of each step in the filter phase and verification phase, respectively. As shown in Figure 7, it illustrates that the execution time consumed in the filter phase is longer than that in the verification phase. Because the computational complexity of partitioning ontologies is lower than the computational complexity of matching ontologies, our method can retain the efficiency. As shown in Figure 8, in the filter phase, 70%–80% entities participate in the calculation; in verification, only 50% entities participate in the calculation. About 20%–30% entities are reduced in the filter phase which also illustrates that our method can retain the efficiency.

7. Conclusions

Aiming at solving the problems existing in large-scale ontology matching, a method of matching large-scale ontologies based on filter and verification was proposed in this paper. The proposed method included two phases: filter phase and verification phase. In the filter phase, ontology was partitioned into several subontologies. It reduced the degree of heterogeneity and scale of ontology. Varieties of irrelevant entities pairs were recognized beforehand, whose similarities were not calculated in subsequent steps. In the verification phase, extracted subontologies were matched and the alignments of each subontology pairs were integrated as the final output. Anatomy and Food were used for evaluation, and the experiments demonstrated that the proposed method improved the recall rate of large-scale ontology matching and retained accuracy rate and efficiency. However, the proposed method also has some drawbacks. Since the original structure of the ontology was changed due to the extraction of subontology in the filtering phase, the accuracy rate was slightly lower. In the future, blank entities will be adopted to replace the original entities to maintain the original structure of ontology and to improve the accuracy rate of large-scale ontology matching.

Data Availability

The Anatomy and Food ontology used in this article are available from OAEI organization (http://oaei.ontologymatching.org).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by The National Key Research and Development Program of China (Grant no. 2018YFB1701703) and Science and Technology Program of Datong City (Grant no. 2019165).

References

[1] J. Euzenat and P. Shvaiko, "Classifications of ontology matching techniques," *Ontology Matching*, Springer, Berlin, Germany, 2013.

[2] M. Cheatham and P. Hitzler, "String similarity metrics for ontology alignment," *Advanced Information Systems Engineering*, Springer, Berlin, Germany, 2013.

[3] M. Pietranik and N. T. Nguyen, "A multi-attribute based framework for ontology aligning," *Neurocomputing*, vol. 146, pp. 276–290, 2014.

[4] S. Hussain, J. D. Roo, and M. C. Jaulent, "Proof-based ontology matching: finding semantic similarities between ancestor graph structures," in *Proceedings of the IEEE Sixth International Conference on Semantic Computing*, Palermo, Italy, September 2012.

[5] V. Mascardi, A. Locoro, and P. Rosso, "Automatic ontology matching via upper ontologies: a systematic evaluation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 609–623, 2010.

[6] P. Arnold and E. Rahm, "Enriching ontology mappings with semantic relations," *Data & Knowledge Engineering*, vol. 93, pp. 1–18, 2014.

[7] M. Mao, Y. Peng, and M. Spring, "An adaptive ontology mapping approach with neural network based constraint satisfaction," *Journal of Web Semantics*, vol. 8, no. 1, pp. 14–25, 2010.

[8] D. Faria, C. Martins, A. Nanavaty et al., "Agreement maker light results for OAEI 2014," *CEUR Workshop Proceedings*, vol. 1317, 2014.

[9] M. Vargas-Vera and M. Nagy, "State of the art on ontology alignment," *Mobile Computing and Wireless Networks*, IGI Global, Philadelphia, PA, USA, 2015.

[10] J. Zhou, J. Liu, and H. Yang, "Dynamic ontology for engineering knowledge management of complex product research & development," *Journal of Computer-Aided Design & Computer Graphics*, vol. 28, no. 11, pp. 1957–1964, 2016.

[11] M. Pietranik and N. T. Nguyen, "A method for ontology alignment based on semantics of attributes," *Cybernetics and Systems*, vol. 43, no. 4, pp. 319–339, 2012.

[12] T. H. Duong and G. S. Jo, "Enhancing performance and accuracy of ontology integration by propagating priorly matchable concepts," *Neurocomputing*, vol. 88, pp. 3–12, 2012.

[13] Y. Wu, S. Yang, and X. Yan, "Ontology-based Subgraph querying," in *Proceedings of the IEEE 29th International Conference on Data Engineering*, Brisbane, Australia, April 2013.

[14] G. Diallo, "An effective method of large scale ontology matching," *Journal of Biomedical Semantics*, vol. 5, no. 1, pp. 44–62, 2014.

[15] W. Hu, Y. Qu, and G. Cheng, "Matching large ontologies: a divide-and-conquer approach," *Data & Knowledge Engineering*, vol. 67, no. 1, pp. 140–160, 2008.

[16] Y. Lin, J. Zhang, Y. Ying et al., "FVBM: A Filter-Verification-Based Method for finding top-K closeness centrality on dynamic social networks," *Web Technologies and Applications*, Springer, Berlin, Germany, 2016.

[17] W. Winkler, *The State of Record Linkage and Current Research Problems, Technical Report, Statistics of Income Division*, Internal Revenue Service Publication, Washington, DC, USA, 1999.

[18] X. Su and J. A. Gulla, "An information retrieval approach to ontology mapping," *Data & Knowledge Engineering*, vol. 58, no. 1, pp. 47–69, 2006.

[19] J. Hayes and C. Gutiérrez, "Bipartite graphs as intermediate model for RDF," in *Proceedings of the 3rd International Semantic Web Conference*, vol. 3298, Springer, Hiroshima, Japan, pp. 47–61, November 2004.

[20] M. Mohammadi, W. Hofman, and Y. Tan, "SANOM results for OAEI 2019," in *Proceedings of the 14th International Workshop on Ontology Matching*, Auckland, New Zealand, October 2019.