2016 IEEE 30th International Conference on Advanced Information Networking and Applications

Leveraging Structural Information in Ontology Matching

Cheng Xie*, Melisachew Wudage Chekol†, Blerina Spahiu‡, Hongming Cai*
*Shanghai Jiao Tong University
{chengxie, hmcai}@sjtu.edu.cn
†University of Mannheim
mel@informatik.uni-mannheim.de
‡University of Milan-Bicocca
spahiu@disco.unimib.it

***Abstract*—Ontology matching is an important part of enabling the semantic web to reach its full potential. Most existing ontology matching methods are mainly based on linguistic information (label, name, title and comment) but from the results achieved it is realized that this information is not sufficient. The latest ontology matching research works are trying to deeply dig into the structural information of ontologies by using "similarity-flooding" method. However, there are several innate issues in similarity-flooding methods that lead to wrong matching results. In this paper, we report the problems of similarity-flooding in ontology matching and propose a novel method to effectively leverage the structural information of the ontology. The evaluation is conducted on OAEI ontology matching benchmarks from 2011 to 2015. The result shows that the proposed approach performs comparatively well with other state of the art matching systems.**

I. Introduction

A. What are the main purposes of ontology matching?

Many traditional challenges in classic applications can be viewed as matching problems such as schema intergration, data warehousing, data integration and catalogue integration [2]. In recent years, applications started to make use of ontologies to describe the semantic of their data. More and more data are being published by different sources making the web a gold mine for people to consume and integrate data. An ontology is referred to a vocabulary that describes a domain of interest and specifies the meaning of terms used in that specific vocabulary [1]. By the increasing usage of ontologies in applications, traditional matching problems are rising to ontology matching problems.

B. What are the challenges?

To understand and integrate data, consumers have to deal with variation in meaning and ambiguity in interpreting it. **Syntactic heterogeneity:** is referred to the usage of different ontology languages, which can be solved by translating them into one language [3]. **Terminological heterogeneity:** is referred to the variations in names (title, label, comment, etc.) when referring to the same entity in different ontologies. e.g. "Abstract" and "Summary" are usually used to describe the same concept in the academic domain.

Semantic heterogeneity: might happen due to the usage of different (and, sometimes, equivalent) axioms for defining concepts or due to the usage of totally different concepts, e.g., in some situations, "Abstract" might be an abstract of a technical paper but "Summary" might be a conclusion of financial report that makes "Abstract" and "Summary" not the same concept anymore.

C. How state of the art systems deal with these challenges?

With the substantial development of ontology models within the last ten years, such as Jena Model [5] and OWL API [4], which accept various ontology representations, syntactic heterogeneity is no longer an unchallengeable concept. Nowdays, state of the art ontology matching systems also handle terminological heterogeneity well by applying various string-based matchers including TF-IDF, Levenstein distance, Jaccard similarity, Wordnet-based matching, etc [6]. In last five years, advanced ontology matching systems, such as AML [7], Logmap [8], DKP-AOM [9], etc. have designed terminological methods and achieve a very high accuracy in ontology matching. However, even with these elaborated methods, semantic heterogeneity cannot be effectively solved since terminological methods cannot directly match semantic heterogeneity. Current ontology matching systems are trying to treat semantic heterogeneity as structural heterogeneity in ontology structure. The idea of the method is to transform an ontology into a graph in which semantic heterogeneity can be represented by structural heterogeneity of the graph. The mainstream method to apply such structural matching is using **similarity flooding** algorithm [10]. From 2011 to 2015, the top 3 matching system, YAM++ [11], LILY [12] and CroMatcher [13], in OEAI ontology matching benchmark test are all using the idea of similarity flooding and obtained a relatively high performance.

However, eventhough applying similarity flooding seems to achieve high performance there is still a problem hiding behind this mainstream method that will be discussed in details in Section III.

II. Preliminaries

RDF Triples: An RDF (Resource Description Framework)[1] triple is a statement that describes the relations between two

[1]http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

objects which is of form $subject \subseteq IRI \cup BlankNode$, $predicate \subseteq IRI$, $object \subseteq IRI \cup Literal \cup BlankNode$. For example, "<*lamb*> <*eats*> <*grass*>" is a RDF triple in which <*lamb*> is the *subject*, <*eats*> is the *predicate* and <*grass*> is the *object*. Formally, IRI[2] is the short name of Internationalized Resource Identifier that is used to uniquely identify an RDF resource. BlankNode[3] is an intermediate node that is used to connect other nodes together. In RDF triples, *Literals*[4] are used for values such as strings, numbers, and dates.

RDF Graph: An RDF graph G is a set of RDF triples and is defined as:

$$G = (N, E), N \subseteq subject \cup object, E \subseteq predicate$$

In an RDF Graph, *subject* and *object* of a triple can be nodes of the graph while the predicate denotes the relationship between them and can be represented by an arrow. An RDF Graph example is shown in Fig. 1 (a).

OWL: Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between them. OWL is based on the basic elements of RDF but adds more vocabulary for descibing web of ontology. Therefore, an OWL graph is actually an RDF graph. An example of OWL can be found in Fig. 1 (b).

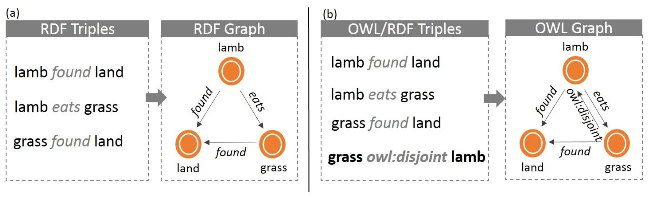


Fig. 1. (a) An example of RDF graph. (b) An Example of OWL graph, here, an OWL predicate *owl:disjoint* is used to declare that *lamb* is different from *grass*.

Similarity-Flooding: Similarity-Flooding is a general graph matching algorithm that is used to calculate similarity between nodes in one or two graphs. It assumes that the similarity between two nodes can be also propagated to their neighbors (adjacent nodes). Furthermore, the more neighbors two nodes have, the less similarity they can propagate to each neighbor. The exact amount of similarity that two nodes can propagate is equal to the inverse number of their neighbors' product, i.e., the propagation is $\frac{1}{N_1 \times N_2}$, where N_1 and N_2 are the neighbors of two nodes respectively. Obviously, similarity propagation is an iterative process since a node pair can only propagate similarity to direct neighbors at a time. For instance, a classic example of similarity flooding is showed in Fig. 2.

III. Problem Statement

Ontology is described in triples which can be connected as a directed graph. The idea of matching semantic heterogeneity

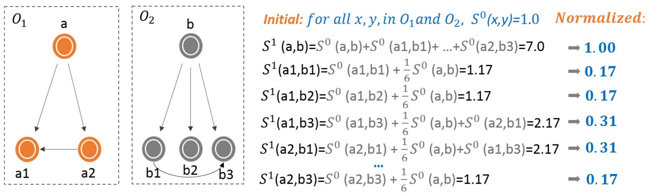

O_1
a
a1
a2
O_2
b
b1
b2
b3
Initial: for all x, y, in O_1 and O_2, S^0(x,y)=1.0
Normalized:
S^1 (a,b)=S^0 (a,b)+S^0 (a1,b1)+ ...+S^0(a2,b3)=7.0 → 1.00
S^1(a1,b1)=S^0 (a1,b1) + $\frac{1}{6}S^0$ (a,b)=1.17 → 0.17
S^1(a1,b2)=S^0 (a1,b2) + $\frac{1}{6}S^0$ (a,b)=1.17 → 0.17
S^1(a1,b3)=S^0 (a1,b3) + $\frac{1}{6}S^0$ (a,b)+S^0 (a2,b1)=2.17 → 0.31
S^1(a2,b1)=S^0 (a2,b1) + $\frac{1}{6}S^0$ (a,b)+S^0 (a1,b3)=2.17 → 0.31
...
S^1(a2,b3)=S^0 (a2,b3) + $\frac{1}{6}S^0$ (a,b)=1.17 → 0.17

Fig. 2. The classic example of similarity flooding. $S^0(x, y)$ means the initial similarity between x and y is always set as 1.0. $S^1(x, y)$ means the similarity after first iteration. O_1 and O_2 are two RDF graphs.

is to apply mature graph matching algorithms on ontology graph. In practice, three state of the art matching systems, YAM++, LILY and CroMatcher use similarity flooding as its core structural matcher and achieve an impresive performance in 2013 and 2015 OAEI ontology matching contest [13].

However, there still exist an innate problem hiding in similarity flooding approach that might lead to wrong matching. **Error Propagation:** Correct mapping pairs propagate simlarity to their neighbors but incorrect mapping pairs propagate wrong similarity to their neighbours. As Fig. 3 shows, *land* and *sea* are asserted as equal due to the incorrect similarity propagated from (*lamb*, *grampus*) and (*grass*, *seal*) which are wrong mappings. In this case, two ontologies describe two different things but share similar graph structure which is constituted by two edges, *eats* and *found*. Indeed, similarity-flooding-based ontology matching would fail if two ontologies share similar graph structure but describe different domains, e.g., an ontology about land animal with an ontology about sea animal.

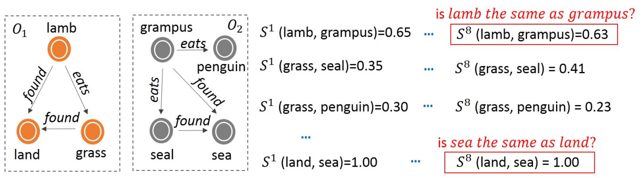

O_1
lamb
found
eats
found
land
grass
grampus
O_2
eats
penguin
eats
found
found
seal
sea
is lamb the same as grampus?
S^1 (lamb, grampus)=0.65 ... S^8 (lamb, grampus)=0.63
S^1 (grass, seal)=0.35 ... S^8 (grass, seal) = 0.41
S^1 (grass, penguin)=0.30 ... S^8 (grass, penguin) = 0.23
...
is sea the same as land?
S^1 (land, sea)=1.00 ... S^8 (land, sea) = 1.00

Fig. 3. An example of error propagation in ontology matching by using similarity flooding. In this case, the calculation reach the fixpoint after eight iterations, marked as S^8

The authors of LILY [12] also report such problem.They try to use more strict constrains and thresholds to prevent errors from one mismatched pair to propagate similarity to their neighbors. However, such constrains and thresholds are hard to set since different evironments or different ontologies would require different constrains and thresholds. Furthermore, ontology matching tasks ususally do not have a training set for threshold learning. YAM++ [11] assumes that error propagation can be eliminated after several iterations but they did not give the evidence to support this assumption.

In this paper, we propose a novel structural matching approach which fundamentally resolve the error propagation in ontology matching. The idea behind our approach is quite simple: if two ontologies can be matched, they must share some concepts. More specifically, similarity should not be

[2]http://www.w3.org/TR/rdf11-concepts/#dfn-iri
[3]http://www.w3.org/TR/rdf11-concepts/#dfn-blank-node
[4]http://www.w3.org/TR/rdf11-concepts/#dfn-literal

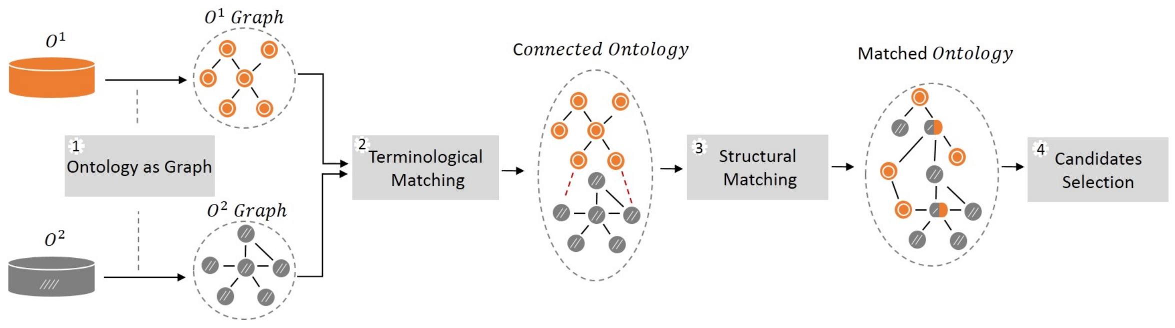


Fig. 4. Overview of the approach. The first step is *Ontology as Graph* which is used to transform an OWL ontology to a proper RDF graph. The second step is *Terminological Matching* which applies various string matchers to connect two ontology in terminological level. *Structural Matching*, the third step, is trying to deeply match the ontologies in structural level. The last step is *Candidates Selection* which is used to select the final mappings.

calculated if two ontology graphs are not intersected. The details of the approach are desrcibed in section V and section VI.

IV. Overview of the Approach

The global pipeline of the approach, as shown in Fig. 4, consists of four processes named *Ontology as Graph*, *Terminological Matching*, *Structural Matching* and *Candidates Selection*.

OWL ontology can be equivalently transformed into RDF triples using OWL-Triple[5]. In order to keep the consistency between OWL-ontology and OWL-triples, OWL-Triple applies BlankNode to express restriction and collection. However, BlankNode lengthen the graph distance between two relative nodes, though hiding their semantics. Thus, the first process *Ontology as Graph* creates a complete ontology graph in which relative nodes are directly connected. The details of OWL ontology transformation are described in Section V.

Reminding that the idea of the approach is that the similarity between ontology nodes can be measured iff two ontology graphs are intersected somewhere (i.e. ontology matching requires at least one shared node between two ontologies). Thus, the second process *Terminological Matching* is designed to discover easy-to-find mappings in order to connect two ontologies as much as possible. Terminological Matching usually applies string-based matchers to match the information about label, name, title, comment and other string information between two ontologies. As string-based matchers can be used Jaccard-Similarity, Levenshtein-Distance, Wordnet-based-Similarity or other metrics. In our approach, Lin-Similarity [19], a wordnet-based similarity metric, and Levenshtein combined with Jaccard-Similarity are applied for Terminological Matching.

Once Terminological Matching process has finished, a connected graph is generated. The graph consists of the two ontologies connected by mapping nodes from Terminological Matching. Then, the Structural Matching process is applied on graph to discover correspondences between two ontologies. In this paper, we propose a novel approach on structural matching that is discussed in section VI in details.

After matching, mapping candidates are listed. In candidates selection process, all candidates are ranked and selected according to one-to-one mapping rule, conspicuousness and similarity threshold.

V. Ontology as Graph

In order to take advantage of the structured information of an ontology, we need to treate it as a graph. As in section IV the first step of the approach is to transform ontology into a graph. Ontologies we deal with are usually OWL ontologies which are based on RDF triples, or RDF graph. In OWL ontology classes, instances, properties, restrictions and statements can be transformed into RDF triples according to OWL-Triples transformation rules.

Though OWL ontology can be completely transformed into RDF graph by OWL-triples transformation rules, there are still two elements in OWL ontology which cannot fit our requirements after transformation. These elements are Restriction and Collection which added at least one BlankNode while building the graph, thus reducing its semantics. Considering the example shown in Fig. 5, a restriction restricts that a *lamb* can eat *grass*. In RDF graph, this restriction is represented as the node *lamb* is a subclass of a BlankNode *"_:a"* which has *owl:someValue* link with *grass*. The BlankNode *"_:a"* contains very weak semantics in this RDF graph because it is used just to hold a position to connect *lamb*, *eats* and *grass* together. Therefore, *lamb*, *eats* and *grass* are indirectly linked while these nodes are supposed to be directly linked instead. To solve this disadvantage, we extend RDF graph by introducing Similarity Graph.

Definition 1: a Similarity Graph G_s is an extended RDF Graph defined as:.

$$\begin{aligned} &G_s = (N, E \cup E_{aux}) \\ &where\ N \subseteq subject \cup object \\ &E, E_{aux} \subseteq predicate, E \cap E_{aux} = \emptyset \end{aligned}$$

Similarity Graph G_s is an extension of OWL/RDF graph by adding auxiliary edges E_{aux} on OWL Restriction and RDF

[5]http://www.w3.org/TR/owl-parsing/

Collection. E_{aux} is a new edges set (i.e., $E \cap E_{aux} = \emptyset$). Edges in E_{aux} hold different semantic meanings from the edges in E. For instance, an auxiliary edge aux:range does not share the same semantics with rdfs:range. In ontology matching, auxiliary edges are treated as different edges with respect to original edges. We create two transformation rules for adding auxiliary edge into OWL graph.

Transformation Rule 1: Restriction: IF N_1 *rdfs:subClassOf* N_2 AND N_2 *rdf:type owl:Restriction* AND N_2 *owl:onProperty* N_3 AND N_2 *condition* N_4, THEN, a new triple N_1 *aux:localname(*N_3*)* N_4 is added. Here, the condition is a set of OWL restriction properties[6] such as *owl:someValueFrom*, *owl:allValueFrom*, *owl:cardinality*, etc. An example of restriction transformation is shown in Fig. 5.

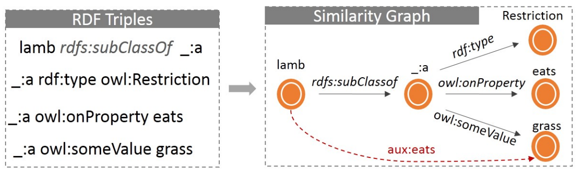


Fig. 5. Ontology Restriction as RDF-Triples. Using auxiliary edge *aux:easts* to directly link *lamb* and *grass*

Transformation Rule 2: RDF Collection. IF N_1 *edge* N_2 AND N_2 *rdf:type rdf:List*, THEN, add new triples N_1 *aux:localname(edge)* (each nodes in N_2). An example of collection transformation can be found in Fig. 6.

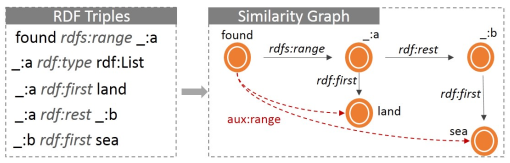


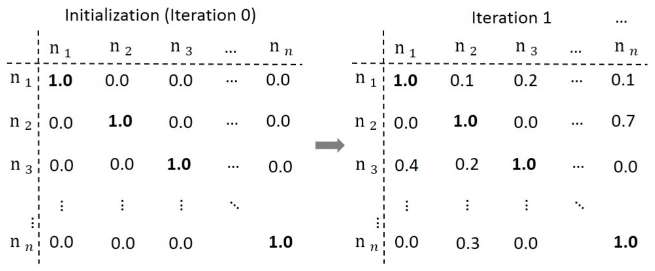
Fig. 6. RDF Collection as RDF-Triples. Using auxiliary edge *aux:range* to directly link *found*, *land* and *sea*

To summarize, an OWL ontology can be represented as an RDF Graph according to OWL-Triple transformation rules. However, Restriction and Collection of OWL ontology are transformed into RDF graph by adding BlankNodes which lengthen the distance between relative nodes. In order to take advantage of the structural information in ontology matching, we extend RDF graph by adding auxiliary edges to directly connect relative nodes on transforming of Restriction and Collection.

VI. Structural Matching

As shown in Fig. 4, the first process of ontology matching is *Ontology as Graph*, which tranforms two ontologies into a proper graph. The second process *Terminological Matching* initially connects two ontologies together by finding easy-to-match pairs between which an *owl:sameAs* link is then added. In this section, we will explain the core method of *Structural Matching*. As discussed, similairity flooding is widely used to discover deep mappings in ontology matching. It implies that similarity can be propagated from one similar pair to their neighbor pairs (adjacent vertex pair). However, the key issue is how to identify similar pairs at first place. In the example in Fig. I, incorrect similairity is propagated due to the wrong initial mapping pairs, where (*Land* and *Sea* are initialized as similar pair and propagate similarity to *Lamb* and *Grampus*).

[6]http://www.w3.org/TR/2004/REC-owl-features-20040210/#s3.4


Initialization (Iteration 0)
Iteration 1
...

	n 1	n 2	n 3	...	n n
n 1	**1.0**	0.0	0.0	...	0.0
n 2	0.0	**1.0**	0.0	...	0.0
n 3	0.0	0.0	**1.0**	...	0.0
⋮	⋮	⋮	⋮	⋱	
n n	0.0	0.0	0.0		**1.0**

	n 1	n 2	n 3	...	n n
n 1	**1.0**	0.1	0.2	...	0.1
n 2	0.0	**1.0**	0.0	...	0.7
n 3	0.4	0.2	**1.0**	...	0.0
⋮	⋮	⋮	⋮	⋱	
n n	0.0	0.3	0.0		**1.0**

Fig. 7. An example of similarity matrix initialization and iteration. Supposing that there are n nodes in two ontologies, a n $\times$ n similarity matrix can be created for all node pairs. The values in the matrix will be updated after each iteration

In our approach, we do not identify similar pairs at first place. Instead, we initialize a similarity matrix for all node pairs in two ontologies before structrual matching. As Fig. 7 shows, each two nodes in the matrix constitute a node pair which has a [0,1] value for measuring their similarity, e.g. $(n_1, n_2) = 0.0$ and $(n_2, n_3) = 0.0$. Specifically, self-pairs, such as (n_1, n_1) and (n_2, n_2), have initially 1.0 similarity score. Obviously, in such similairity matrix, only self-pairs can propagate similarity to their neighbors that guarantes corrected propagations. In structural matching, the similairity calculation is conducted in an iterative way such that the similairity values will be updated after each iteration.

A. Decompositing Similarity Graph by edges

Suppose that the second process *Terminological Matching* finds an easy-to-match pair O_1*:Mammal* and O_2*:Mammal* in Fig. 3 that connects two ontologies together. Then, the RDF Graph in Fig. 3 can be transformed into Fig. 8 (a). The ontology graph in Fig. 8 (a) consists of three edges that are *type*, *eats* and *found*. Each edge hold its own semantics thus we cannot just treat them as only one edge. Therefore, for each type of edge, a sub-graph is built.

Definition 2. Sub Similarity Graph G'_s is a similarity graph set that each similarity graph in the set only contains edges that have same name.

$$G'_s = (N_s, E_s), E_s \in E \cup E_{aux}, N_s \in N$$

For each N_s in G'_s, there is at least one E_s link to N_s. For instance, a Sub Similarity Graph G'_{type} is showed in Fig. 8 (b) that is built on the edge *type*. In the same way G'_{eats} and G'_{found} can be built as shown in Fig. 8 (c) and Fig. 8 (d).

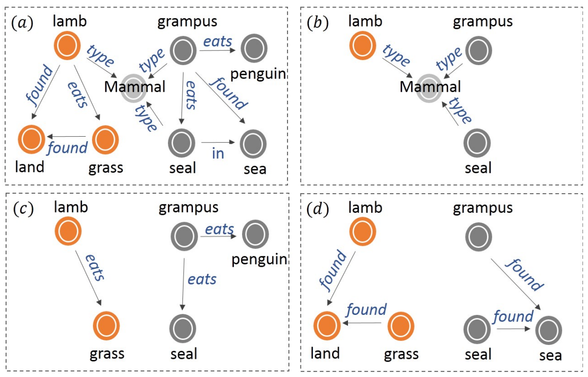


Fig. 8. (a) Two ontologies are connected on the node pair (Mammal,Mammal). (b) The sub graph G'_{type} on the edge *type*. (c) The sub graph G'_{eats} on the edge *eats*. (d) The sub graph G'_{found} on the edge *found*.

B. Calculating similarity on Sub Similarity Graph

Let $S^k(a,b)$ denote the similarity between node a and b at the k^{th} iteration of the complete ontology graph. $N_s(a)$ and $N_s(b)$ denote the neighbors of a and b in Sub Similarity Graph G'_s respectively. The similarity between a and b at the $(k+1)^{th}$ iteration in this sub graph can be calculated by averaging all their neighbor pairs' similarity, marked as $S_s^{k+1}(a,b)$. The formal expression is addressed in Equation 1.

$$S_s^{k+1}(a,b) = \frac{1}{|N_s(a)||N_s(b)|} \sum_{i}^{N_s(a)} \sum_{j}^{N_s(b)} w_s(i,j) S^k(i,j) \quad (1)$$

In Equation 1, $S^k(i,j)$ denotes the aggregated similarity in ontology graph. Similarity aggregation will be introduced in next sub section. $w(i,j)$ denotes the weight of similarity contributed by any two neighbour nodes i and j of a and b. The idea of weight is that the more edges are linked to these two nodes, the less weight they obtain. Therefore, the weight of (i,j) is inversely proportional to the number of neighbors they have. The formal expression of weight w is defined in Equation 2.

$$w(i,j) = \begin{cases} \frac{2}{|e(i)|\cdot(|e(i)|-1)} & i = j \\ \frac{2}{(|e(i)\cup e(j)|)\cdot(|e(i)\cup e(j)|-1)} & i \neq j \end{cases} \quad (2)$$

In Equation 2, $|e(i)|$ denotes the number of edges that are linked to i and $|e(j)|$ denotes the number of edges that are linked to j. If $i \neq j$ the weight is inversely proportional to the number of combination pairs in $e(i) \cup e(j)$, otherwise, the weight of (i,j) is the inverse proportion of the number of combination pairs in $e(i)$.

In practice, the direction of the edges also needs to be taken into account. For each edge on each pair (a,b), in-degree and out-degree should be calculated separately using Equation 1. Then, the average value on in-degree and out-degree is calculated as the similarity value of the pair in this sub graph.

C. Aggregating similarity from sub graphs

At the end of each iteration, similarity will be aggregated from all sub graphs. There are many aggregation strategies including arithmetic or harmonic mean, maximum selection, weighted mean and others. We tried different strategies in pratice and found out that applying arithmetic mean achieves the best performance. The formula of similarity aggregation is addressed in Equation 3.

$$S^k(a,b) = \frac{1}{|G'_s|} \sum_{s}^{G'_s} S_s^k(a,b) \quad (3)$$

In Equation 3, the similarity $S^k(a,b)$ of a and b in complete ontology graph is the average value of their similarity values in all sub graphs while G'_s is the set of all sub graphs.

D. Iteration Fixpoint

Fixpoint is a final status that indicates that iteration and similarity are stable. In the approach, we compare the results of current iteration to the results of the previous iteration. If nothing has changed, then we stop iterating, else we continue to the next one. In some cases, no fixpoints can be reached due to the loop structure of the ontology graph. Thus, a maximum iteration number should be set. In our experiment, the maximum iteration number is set as 50.

E. Summary of structural matching

In order to express the core method more clearly, we address the key-part of structural matching algorithm into pseudocode in Algorithm 1.

Algorithm 1 Structural Matching

```
Input: Similarity Graph $G_s$ and Sub Similarity Graph $G'_s$
Output: Similairity Maritx $M$ .
 1: while $M_{current} \neq M_{last}$ and iteration < Maximum do
 2:     $M_{last} \leftarrow M_{current}$
 3:     $M_{current} \leftarrow 0$ Matrix
 4:     for each node $a$ in $G_s$ do
 5:         for each node $b$ in $G_s$ do
 6:             $sum \leftarrow 0$
 7:             for each sub graph in $G'_s$ do
 8:                 $sum \leftarrow S_s(a,b) + sum$
 9:             end for
10:             $M_{current}(a,b) \leftarrow \frac{sum}{|G'_s|}$
11:         end for
12:     end for
13: end while
14: return $M_{current}$
```

We apply our approach using similarity flooding on the ontology in Fig. 8 (a). The results for each iteration are listed in Table 1. The approach properly leverages the structural information in ontology matching. From the results of Table I, it is shown that error propagation has been effectively prevented. Dissimilar pairs such as (*lamb*, *grampus*) and (*land*, *sea*) are ranked properly with low similarity score, e.g., 0.116

and 0.020 compare to 0.732 and 1.0 in similarity flooding. Furthermore, the similarity gradient is more reasonable than it is in similarit flooding, e.g., (*lamb*, *grampus*) has relatively higher similarity score than (*grass*, *seal*) which itself has higher similarity score than (*grass*, *sea*). In summary, the approach can generate proper similarity score and reasonable similarity gradient. In practice, proper similarity score facilitates matching system to discover potential mapping pairs more broadly. Reasonable similarity gradient helps matching system to remove incorrect mapping pairs more accurately.

TABLE I
SIMILARITY CALCULATION ON EACH ITERATION OF FIGURE 8 (A)

Pair	S^1	S^2	S^3	S^4	Sim-Flooding
(lamb,grampus)	0.111	0.111	0.116	0.116	0.732
(lamb,seal)	0.111	0.111	0.112	0.112	0.359
(lamb,penguin)	0.0	0.0	0.0	0.0	0.0
(grass,seal)	0.0	0.012	0.013	0.014	0.379
(grass,penguin)	0.0	0.012	0.012	0.013	0.225
(land,sea)	0.0	0.018	0.019	0.020	1.0
(Mammal,Mammal)	1.0	1.0	1.0	1.0	1.0

VII. EVALUATION

In this section, we conduct several experimental studies. Firstly, we provide several representative examples to give a subjective expression on the ontology matching. Then, we compare the approach with the state of the art matching systems on OEAI bechmark dataset from 2011 to 2015. At last, we discuss the proper threshold for the approach.

A. Dataset preparation

The dataset we use is selected from OAEI benchmark[7]. and it is created from a bibliographic ontology which we call *source ontology*. From the source ontology, 106 corresponding ontologies are built by applying different tranformation rules[8] such as random string, synonyms name, different conventions, other languages, expansed hierarchy, suppressed properties, etc. To match the transformed ontology with the reference one we need to match 106 tasks which have non-sequential number from 101 to 266, and evaluate the results on the corresponding reference.

According to the transformation rules, the dataset can be divided into 4 groups: (1) 101-104: in this group the ontology is compared to a totally irrelevant one, also language generalization and ontology restriction are applied. (2) 201-210: the ontology structure is preserved, but the textual information is partially suppressed. (3) 221-247: the textual information of the ontology is preserved, but the structure of the information is partially changed. (4) 248-266: This is the most difficult test set. The textual information of the ontology is suppressed, and also the structure of the information is partially changed.

[7]http://oaei.ontologymatching.org
[8]Transformation Rule: http://oaei.ontologymatching.org/tests/

B. Result and comparison

To evaluate the performance of our approach we used precision, recall and F-measure. Let R_1 be the real matching result of the approach and R_2 be the reference result, then the precision, recall and F-measure are calculated as defined in Equation (4).

$$\begin{aligned} Precision &= \frac{|R_1 \cap R_2|}{|R_1|}, Recall = \frac{|R_1 \cap R_2|}{|R_2|} \\ F{-}measure &= \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \end{aligned} \tag{4}$$

Based on four groups, we first conduct the full experiments on OAEI benchmark 2014 (101-266, including all sub-tasks) and compare with 11 state of the art matching systems using the same data environment. The results are shown in Table II.

TABLE II
EVALUATION ON OAEI BECHMARK BIBLIO ONTOLOGY 2014 (FULL TEST CASES).

System	Group-1 101-104 Prec Rec F	Group-2 201-210 Prec Rec F	Group-3 221-247 Prec Rec F	Group-4 248-266 Prec Rec F
baseline	0.64 1.00 0.78	0.26 0.41 0.32	0.75 1.00 0.86	0.28 0.39 0.33
AOT_2014	0.97 0.97 0.97	0.77 0.66 0.71	0.99 0.98 0.98	0.54 0.39 0.45
AOTL	0.96 0.96 0.96	0.77 0.66 0.71	0.98 0.98 0.98	0.57 0.38 0.45
LogMapLite	0.56 0.99 0.72	0.30 0.39 0.34	0.60 1.00 0.75	0.35 0.39 0.37
LogMap-C	0.58 0.96 0.72	0.30 0.36 0.32	0.55 0.86 0.67	0.32 0.31 0.31
LogMap	0.56 0.96 0.7	0.29 0.36 0.32	0.50 0.86 0.63	0.30 0.31 0.30
MaasMtch	**1.00 1.00 1.00**	**0.99 0.61 0.75**	0.71 0.26 0.38	0.25 0.16 0.19
AML	0.00 0.00 0.00	0.71 0.26 0.38	0.95 0.95 0.95	0.69 0.32 0.44
XMap2	**1.00 1.00 1.00**	0.80 0.26 0.39	1.00 0.96 0.98	0.78 0.31 0.44
RSDLWB	**1.00 1.00 1.00**	0.79 0.39 0.52	**1.00 1.00 1.00**	**0.77 0.39 0.52**
OMReasoner	0.87 1.00 0.93	0.60 0.40 0.48	0.80 1.00 0.88	0.51 0.39 0.44
This Paper	1.00 1.00 1.00	0.99 0.86 0.92	0.99 1.00 0.99	0.96 0.68 0.79

In Table II, the baseline is a pure string-based matching system which is used to show the base performance on this dataset. From the results we can see that current matching systems have all outpeformed baseline. Especially, in Group-1 (101-104) and Group-3 (221-247), most of the systems obtain very high performance and some of them even get 100% F-measure, such as RSDLWB [22] and XMap2 [22]. Therefore, it can be inferred that state of the art matching systems perform quite well on terminological matching since Group-1 and Group-3 did not suppress textual information in ontologies. However, in Group-2(201-210), the average performance of all systems is relatively lower than the performance of Group-1 and Group-3. In contrary, our approach remains stable and gets a good result (F-measure is **24**% higher than the best system in Group-2). More prominent, compared to the systems in Group-4 which suppress almost all textual information, our approach gets a relatively good result(F-measure is **57**% higher than the best system in Group-4).

In order to clearly show the importance of structural information in ontology matching, we remove all sub-tasks (e.g. 248-2, 248-4, etc.) from Group-2 and Group-4 since sub-tasks contain richer textual information while the main tasks (e.g. 248, 249, 250, etc.) do not. The new results are shown in Table III.

TABLE III
EVALUATION ON OAEI BECHMARK BIBLIO ONTOLOGY 2014 (NO SUB-TASKS).

System	Group-1			Group-2			Group-3			Group-4		
	101-104			201-210			221-247			248-266		
	Prec	Rec	F	Prec	Rec	F	Prec	Rec	F	Prec	Rec	F
baseline	0.64	1.00	0.78	0.02	0.03	0.02	0.75	1.00	0.86	0.01	0.02	0.01
AOT_2014	0.97	0.97	0.97	0.48	0.45	0.46	0.99	0.98	0.98	0.00	0.00	0.00
AOTL	0.96	0.96	0.96	0.49	0.45	0.47	0.98	0.98	0.98	0.00	0.00	0.00
LogMapLite	0.56	0.99	0.71	0.00	0.00	0.00	0.60	1.00	0.75	0.00	0.00	0.00
LogMap-C	0.58	0.96	0.72	0.00	0.00	0.00	0.55	0.86	0.67	0.00	0.00	0.00
LogMap	0.56	0.96	0.70	0.00	0.00	0.00	0.50	0.86	0.63	0.00	0.00	0.00
MaasMtch	**1.00**	**1.00**	**1.00**	**1.00**	**0.36**	**0.53**	0.71	0.26	0.38	**0.17**	**0.06**	**0.09**
AML	0.00	0.00	0.00	0.00	0.00	0.00	0.95	0.95	0.95	0.00	0.00	0.00
XMap2	**1.00**	**1.00**	**1.00**	0.00	0.00	0.00	1.00	0.96	0.98	0.00	0.00	0.00
RSDLWB	**1.00**	**1.00**	**1.00**	0.00	0.00	0.00	**1.00**	**1.00**	**1.00**	0.00	0.00	0.00
OMReasoner	0.87	1.00	0.93	0.00	0.00	0.00	0.80	1.00	0.89	0.00	0.00	0.00
This Paper	1.00	1.00	1.00	0.98	0.85	0.91	0.99	1.00	0.99	0.90	0.38	0.53

As the results show in Table III, almost all systems fail to match the ontologies in Group-2 and Group-4 when almost all textual information is removed. In Group-2, except of baseline, only three systems, AOT_2014, AOTL and MaasMtch, could get nonzero result compare to only one system with nonzero result in Group-4. In contrary, our approach still gets a good result in Group-2 (0.98 precision and 0.85 recall) and a relatively good result in Group-4 (0.90 precision and 0.38 recall). Comparing to other systems in OAEI 2014 bechmark test, we get **72**% increase on F-measure in Group-2. In Group-4, in which all textual information are removed, it is hard to compare the results since almost all other systems cannot find any matchings.

For a more complete evaluation, we also conduct four experiments on OAEI benchmark dataset from 2011 to 2014. The results are shown in Table IV.

TABLE IV
EVALUATION ON OAEI BECHMARK BIBLIO ONTOLOGY. COMPARISON WITH TOP-15 MATCHING SYSTEMS FROM 2011 TO 2014

System	2011 F-meas	2012 F-meas	2013 F-meas	2014 F-meas
YAM++	**0.74**	**0.83**	**0.89**	-
CroMatcher	-	-	0.88	-
AROMA	-	0.77	-	-
CIDER-CL	-	0.75	-	-
IAMA	-	0.73	-	-
CODI	0.73	-	-	-
CSA	0.72	-	-	-
ODGOMS	-	-	0.71	-
AUTOMSv2	-	0.69	-	-
Wikimatch	-	0.62	0.69	-
MaasMtch	-	0.60	0.069	0.56
Hertuda	-	0.68	0.68	-
Hotmatch	-	0.66	0.68	-
RSDLWB	-	-	-	**0.66**
AOTL	-	-	-	0.65
This Paper	-	0.88	0.88	0.88

We could not obtain consistent results in 2011 with other systems since the dataset published on the site today is different from the dataset that have been used in the competition. However, from 2012, the dataset of the benchmark becomes stable, meaning that if a system gets a result in 2014, it is likely to get the same or similar result in 2013, and 2012. As the results in Table IV shows, the best matching system YAM++, started from 2011, keeps progressing each year (from 0.74 in 2011 to 0.89 in 2013). It is not only an effective ontology matching approach but also a well designed matching system. Similar to YAM++, CroMatcher also gets a good result (0.88) in 2013. Although YAM++ and CroMatcher have good performance, our approach can achieve comparable result with them. Furthermore, since the paper is aimed to show how structural information helps ontology matching and how to correctly leverage structural information in ontology matching, it is not yet a matching system.

C. Summary of the experiment

The results from four experiments on OAEI benchmark from 2011 to 2014 show that the proposed approach is a good method to be applied in ontology matching. From the result of Group-1 and Group-3 in Table 2 and Table 3, our approach can achieve comparable performance with other state of the art matching systems. Significantly, on matching structural ontologies (Group-2 and Group-4 in Table II and Table III), the approach significantly outperforms other matching systems. As showed in Table IV, compare to all ontology matching systems from OAEI-2011 to OAEI-2014, our approach achieves comparatively high performance on the benchmark test (0.88 F-measure).

VIII. RELATED WORKS

As discussed in the paper, there are two main approaches in ontology matching classified by the orientation of the ontologies. Rich-text ontologies require more effort on string matching, such as biomedical ontologies[9] and anatomy ontology[10]. In the contrary, rich-structure ontologies require more effort on structure matching such as Biblio[11] and DogOnto[12]. Note that string-based and structure-based methods are used together in ontology matching systems.

A. String-based ontology matching

String-based ontology matching methods are summarized in [14]. In this paper it is also shown how well can pure string matchers perform in ontology matching. From the results pure string-based methods achieve comparable results with state of the art ontology matching systems applied in rich-text ontologies (anatomy and multifarm ontology). In general, the classic of pure string-based methods include Jaccard, Levenstein, TF-IDF [6], Jaro-Winkler [21] and WordNet series (e.g. in [15] wup measure calculates relatedness by considering the depth of two synsets in WordNet taxonomies) are widely used in state of the art ontology matching systems. AROMA [16] is a well designed matching system based on Jaro-Winkler string matcher. Since it mainly relies on textual information, it cannot obtain good recall when alteration affects text annotation both in class/property description and

[9]http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2014/
[10]http://oaei.ontologymatching.org/2014/anatomy/anatomy-dataset.zip
[11]http://oaei.ontologymatching.org/tests/101/onto.rdf
[12]http://elite.polito.it/ontologies/dogont.owl

in their individual/property values in ontologies. CODI [17] combines Cosine, Levenshtein, Jaro Winkler, Simth Waterman Goto, Overlap Coefficient and Jaccard similarity measures with specific weights that guarantee high precision in ontology matching. Innovatively, SILK [18] applies an unsupervised algorithm to automatically combine all above string-based matchers to obtain good results in different ontology matching environments. Even so, both CODI and SILK cannot obtain good recall matching ontologies where textual information is rare.

B. Structure-based ontology matching

Similarity-Flooding [10] and Simrank [20] are two basic and widely used structural matching approaches. The former assumes that the similarity between two nodes can be propagated to their neighbors by shared links. But the latter realizes that two nodes are similar if all their neighbors are similar. As we mentioned in section II, error propagation is an inevitable feature of similarity flooding since errors can be also propagated when similarities are propagating. Nevertheless, after many improvements, designing and parameters learning, a few but not many state of the art matching systems based on similarity flooding have already achieve a good performance in ontology matching where textual information is rare. Representatively, YAM++ [11] and LILY [12] are two best matching systems in OAEI benchmark test which implement similarity flooding as their core matcher. By applying more strict constrains and thresholds in specific datasets, error propagation can be prevented. However, it is hard to dynamically set proper constrains and thresholds for generic usage.

IX. Conclusion

Ontology matching is an important part of enabling the semantic web to reach its full potential. There are two primary challenges in ontology matching called terminological heterogeneity and structural heterogeneity. State of the art matching systems almost solved the terminological heterogeneity. However, semantic heterogeneity which is cased by the heterogenous structure of ontology still remains a challenge to the current matching systems. In the paper, we proposed a novel structural matching approach which is focused on solving structure heterogeneity in ontology matching. By matching two ontologies on terminological level and only set similarity to self-pairs before structural matching, the error propagation that appears in normal structural matching has been effectively prevented. By aggregating structural similarity from Graph Slices of ontology, structural information of ontology are leveraged to facilitate ontology matching. The first try of the approach on OEAI benchmark test (2011 to 2014) shows that the approach is quite effective in dealing with structural heterogeneity in ontology matching.

Acknowledgment

The paper have been supported in part by National Natural Science Foundation of China under contract number 71171132, and 61373030. We would like to thank Gu Yizhi for helping running the experiments.

References

[1] P. Shvaiko and J. Euzenat, *Ontology matching: state of the art and future challenges*, IEEE Transactions on Knowledge and Data Engineering (2013).

[2] J. Euzenat and P. Shvaiko, *Ontology matching*, Springer, 2007.

[3] J. Euzenat and H. Stuckenschmidt, *The 'Family of Languages' Approach to Semantic Interoperability*, Knowledge Transformation for the Semantic Web (2003): 49-63.

[4] M. Horridge and S. Bechhofer *The OWL API: A Java API for OWL ontologies.*, Semantic Web (2011): Volume 2, number 1, 11-21, .

[5] J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson *Jena: implementing the semantic web recommendations*, Proceedings of the 13th international World Wide Web conference on Alternate track papers (2004): 74-83.

[6] W. Cohen, P. Ravikumar and S. Fienberg *A comparison of string metrics for matching names and records*, Kdd workshop on data cleaning and object consolidation, Volume 3, pages 73–78, 2003.

[7] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. Cruz, and F. M. Couto, *The agreementmakerlight ontology matching system*, On the Move to Meaningful Internet Systems: OTM 2013 Conferences, pages 527–541, 2013.

[8] E. Jiménez-Ruiz, and B. C. Grau, *Logmap: Logic-based and scalable ontology matching*, The Semantic Web–ISWC (2011): 273-288.

[9] M. Fahad, M. Nejib, and A. Bouras. *Detection and resolution of semantic inconsistency and redundancy in an automatic ontology merging system.* Journal of Intelligent Information Systems (2012): 535-557.

[10] S. Melnik, H. G. Molina, and E. Rahm. *Similarity flooding: A versatile graph matching algorithm and its application to schema matching.* 18th International Conference on Data Engineering, Proceedings (2002).

[11] N. DuyHoa, and Z. Bellahsene. *YAM++: A multi-strategy based approach for ontology matching task.*, Knowledge Engineering and Knowledge Management. Springer Berlin Heidelberg (2012): 421-425.

[12] W. Peng, and B. Xu. *Lily: Ontology alignment results for oaei 2008.*, Proceedings of the Third International Workshop on Ontology Matching(2008).

[13] B. C. Grau, Z. Dragisic, K. Eckert, J. Euzenat, A. Ferrara, R. Granada, V. Ivanova, E. Jimenez-Ruiz, A. Kempf, P. Lambrix and O. Zamazal. *Results of the ontology alignment evaluation initiative 2013.*, International Workshop on Ontology Matching, collocated with the 12th International Semantic Web Conference(2013).

[14] M. Cheatham and P. Hitzler. *The properties of property alignment.*, Proceedings of the 9th International Workshop on Ontology Matching collocated with the 13th International Semantic Web Conference (2014): 13-24.

[15] Z. Wu and M. Palmer. *Verb semantics and lexical selection.*, In Proceedings of the 32nd Annual meeting of the Associations for Computational Linguistics (1994): 133-138.

[16] J. David and F. Guillet *Association Rule Ontology Matching Approach.*, International Journal on Semantic Web and Information Systems 3.2 (2007): 27-49.

[17] J. Huber, T. Sztyler, J. Noessner, and C. Meilicke. *CODI: Combinatorial Optimization for Data Integration Results for OAEI 2011.*, Ontology Matching 134 (2011).

[18] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. *Silk-A Link Discovery Framework for the Web of Data.*, LDOW (2009), 538.

[19] Lin, Dekang. *An information-theoretic definition of similarity.*, ICML. Vol. 98. 1998.

[20] G. Jeh and J. Widom. *SimRank: a measure of structural-context similarity.*, In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (2002): 538-543.

[21] W. E. Winkler. *The state of record linkage and current research problems.*, Statistics of Income Division, Internal Revenue Service Publication R99/04 (1999).

[22] Dragisic, Zlatan, et. al *Results of the ontology alignment evaluation initiative 2014*. Proceedings of the 9th International Workshop on Ontology Matching Collocated with the 13th International Semantic Web Conference (ISWC 2014). 2014.