Matching Weak Informative Ontologies

Peng Wang^a

^aSchool of Computer Science and Engineering, Southeast University, Nanjing 210096, China

Abstract

Most existing ontology matching methods utilize the literal information to discover alignments. However, some literal information in ontologies may be opaque and some ontologies may not have sufficient literal information. In this paper, these ontologies are named as weak informative ontologies (WIOs) and it is challenging for existing methods to matching WIOs. On one hand, string-based and linguistic-based matching methods cannot work well for WIOs. On the other hand, some matching methods use external resources to improve their performance, but collecting and processing external resources is still time-consuming. To address this issue, this paper proposes a practical method for matching WIOs by employing the ontology structure information to discover alignments. First, the semantic subgraphs are extracted from the ontology graph to capture the precise meanings of ontology elements. Then, a new similarity propagation model is designed for matching WIOs. Meanwhile, in order to avoid meaningless propagation, the similarity propagation is constrained by semantic subgraphs and other conditions. Consequently, the similarity propagation model ensures a balance between efficiency and quality during matching. Finally, the similarity propagation model uses a few credible alignments as seeds to find more alignments, and some useful strategies are adopted to improve the performance. This matching method for WIOs has been implemented in the ontology matching system Lily. Experimental results on public OAEI benchmark datasets demonstrate that Lily significantly outperforms most of the state-of-the-art works in both WIO matching tasks and general ontology matching tasks. In particular, Lily increases the recall by a large margin, while it still obtains high precision of matching results.

Keywords: ontology matching, weak informative ontology, similarity propagation, semantic subgraph

1. Introduction

More and more ontologies are created and used distributively by different communities in the past few decades. Ontology users or engineers would integrate or process multiple ontologies in practical applications. However, ontologies themselves could be heterogeneous.

Email address: pwang@seu.edu.cn (Peng Wang)

Preprint submitted to SCIENCE CHINA: Information Sciences

It is necessary to integrate various ontologies and enable cooperation between them. Ontology matching, which discovers alignments between ontologies, aims to provide a common layer from which heterogeneous ontologies could exchange information in semantically sound manners [1, 2].

Many ontology matching methods have been proposed [3, 4, 5, 6, 7, 8, 9]. Generally, calculating the literal similarity is the most popular matching technique. However, not all ontologies provide sufficient, clear, and precise literal information for describing the semantics of elements in ontologies. For example, in the medical domain, the adult mouse anatomy ontology¹ uses unique codes like MA_0000436 to name concepts, and every term in geneontology² has a unique seven digit identifier GO ID like GO:0006915. In many ontologies, we also notice that some elements have no enough comments or labels to help users to understand their meanings. For another example, the 248-266 ontologies in the OAEI (Ontology Alignment Evaluation Initiative)³ benchmark dataset are such special cases, in which most labels are meaningless strings and even do not have any comment. Actually, these examples are not rare in real-world ontologies, and users still usually meet ontology elements with opaque labels or lack of sufficient information that cannot help users to understand the elements. In some application domains such as semantic Web of Things [10], electric power grid [11, 12] and Industry 4.0 [13], ontologies often miss lexical layer instead of lots of human-unreadable IP addresses, linked sensors, electric switches, machine tools and products. This paper calls an ontology without sufficient or clear literal information the weak informative ontology (WIO).

For this situation, string-based and linguistic-based matching methods cannot work well and will miss a lot of alignments, which leads to low recall of matching results. Therefore, it is necessary to find a feasible solution to solve the problem of matching weak informative ontologies. Utilizing the structure information is a natural idea to compensate for stringbased and linguistic-based methods. Here, the structure-based ontology matching does not mean matching geometrical graphs, which cannot deal with the semantic information in ontologies. Namely, its matching results would have no meaning in semantics. Therefore, traditional graph matching algorithms [14] are not suitable here.

Most structure-based ontology matching methods [15, 16] usually employ the similarity propagation idea [17, 18]: *similar objects are related to similar objects*, which assures that more alignments can be found by providing few alignments as seeds. Several similarity propagation models have been proposed for matching database schemas or XML schemas [17, 18, 19, 20]. However, these algorithms cannot be used for ontology matching problem directly. For example, Blondel's graph matching algorithm [19] cannot obtain good alignments for matching ontologies, because this algorithm directly treats an ontology as a graph and ignores the semantic information in the ontology.

To address the issue of matching WIOs, this paper attempts to propose a practical matching method which is inspired by our previous work [21]. First, we generalize and redefine

¹http://web.informatik.uni-mannheim.de/oaei/anatomy11/index.html

²http://geneontology.org/

³http://oaei.ontologymatching.org

the problem as the weak informative ontology matching problem. Second, we introduce the semantic subgraph to describe elements in ontologies, which is the foundation of our solution, and present the algorithm for extracting semantic subgraphs. Third, we address how to use semantic subgraphs to build a matcher, which can provide credible seeds for subsequent similarity propagation. Fourth, we focus on discussing the new similarity propagation model for ontology matching and corresponding similarity propagation strategies. Finally, we provide more comprehensive experimental results to demonstrate the performance of our method.

The main original contributions of this paper are as follows. (1) This paper proposes semantic subgraphs to capture precise meanings of ontology elements. (2) A new similarity propagation model based on semantic subgraphs is proposed for matching weak informative ontologies. This is a novel similarity propagation model for matching ontologies. The propagation conditions are not only strict but also reasonable for ontology characteristics. In addition, the similarity propagation is constrained by semantic subgraphs, that can avoid meaningless propagation and improve the matching performance. Therefore, compared to the classical general similarity propagation model Similarity Flooding [17], this model is more specific and ensures the balance between matching efficiency and quality. (3) Based on the semantic subgraphs, a matcher using semantic description documents is proposed, and it provides few credible alignments as seeds to the similarity propagation model. Moreover, the similarity propagation model adopts some useful strategies to improve the matching performance. (4) Experimental results show that our method performs well not only for matching weak informative ontologies but also for matching general ontologies. Especially, our method increases the recall by a large margin while still obtains high precision of matching results.

The remainder of this paper is organized as follows: Section 2 addresses the weak informative ontology matching problem. Section 3 presents an overview of the proposed method. Section 4 discusses the ontology graph and its processing. Section 5 introduces the basic principle of semantic subgraphs. Section 6 details the method for extracting semantic subgraphs from ontology graphs. Section 7 discusses the similarity propagation model based on semantic subgraphs. Section 8 analyzes and compares different propagation scale strategies. Section 9 describes the matcher using semantic subgraphs. Section 10 presents the experimental results and corresponding discussions. Section 11 is an overview of related work and Section 12 is the conclusion.

2. Problem Statement and Analysis

Usually, an ontology contains concepts, properties, instances and axioms. We follow the work in [6, 7] and give a formal definition for the ontology matching problem.

Definition 1 (*Ontology Matching*). The matching between two ontologies O_1 and O_2 is: $\mathcal{M} = \{m_k | m_k = \langle e_i, e_j, r, s \rangle\}$, where \mathcal{M} is an alignment; m_k denotes a correspondence with a tuple $\langle e_i, e_j, r, s \rangle$; e_i and e_j represent the expressions which are composed of elements from O_1 and O_2 respectively; r is the semantic relation between e_i and e_j , and rcould be equivalence (=), generic/specific (\exists/\sqsubseteq), disjoint (\bot) and overlap (\Box), etc.; s is the confidence about an alignment and typically in [0, 1] range. Therefore, an alignment \mathcal{M} is a set of correspondences m_k .

This paper only focuses on correspondences of concept - concept and property - property with equivalence relation.

If we have enough and clear literal information about all elements, alignments can be discovered easily. However, real-world ontologies cannot always provide such ideal literal information. Figure 1 shows a matching scenario in OAEI. All concepts of ontology A have necessary comments or labels. However, in ontology B, some elements, such as *izxnquo* and *zdqssqdb*, have meaningless labels and there are no comments to explain these concepts. In this matching scenario, for some elements like A:Conference and B:ScientificMeeting, we can easily calculate matching similarities by linguistic-based or string-based techniques. However, we cannot calculate the similarities for element pairs which contain opaque elements. For example, the similarity between A:Institution and B:izxnquo is difficult to be calculated. Such opaque elements appear in ontologies for two possible reasons. First, ontology engineers do not provide sufficient annotations or comments for all elements, that will make some elements lack enough information to be understood. Second, to name elements, ontology engineers would use some special codes, which cannot be understood by others. For instance, concept Address may be named as Add, Adr or Dizhi (in Chinese spelling). We call such elements the weak informative elements and define as follows.



Figure 1: An weak informative ontology matching scenario

Definition 2 (*Weak Informative Element*). Given an ontology element e and a lexicon \mathcal{L} , let D_e be the set of words in e's literal information including identities, labels, comments and annotations. $|D_e \cap \mathcal{L}|$ denotes the number of words contained in both \mathcal{L} and D_e . D_e can not be \emptyset and $|D_e| \geq 1$. If $\frac{|D_e \cap \mathcal{L}|}{|D_e|} < \phi$, e is a weak informative element, where ϕ is a small threshold value in [0,0.5].

In this paper, ϕ is set to 0.25, and the lexicon \mathcal{L} is WordNet⁴. For a weak informative element e, its meaning is difficult to be understood according to corresponding literal information.

Therefore, if an ontology contains a certain proportion weak informative elements, it is difficult to be understood. We call such an ontology the weak informative ontology and define as follows.

Definition 3 (*Weak Informative Ontology*). Given an ontology O with N elements, let w be the number of weak informative elements and δ be a predefined threshold in [0, 1], if $\frac{w}{N} > \delta$, O is a weak informative ontology (WIO).

For the reason that building an ontology is a time-consuming and error-prone process, an ontology would contain some weak informative elements, which make matching methods difficult to find alignments. Table 1 shows a survey of weak informative ontologies for 10 ontologies in OAEI2008 benchmark dataset ⁵. We manually examine these ontologies and count the number of weak informative elements in each ontology. Column 2 to column 4 are the ratio of weak informative elements in concepts, properties and instances, respectively. The w/N value is the total ratio between number of weak informative elements and total number of elements. The predefined threshold $\delta = 0.25$, that means if the total ratio is larger than 0.25, users cannot understand the ontology well. Consequently, 7 ontologies in Table 1 are weak informative ontologies. The last column shows matching results (average, best and worst F1-measure) of 6 classic systems [22] (Aflood, AROMA, ASMOV, DSSim, GeroMe and MapPSO) and 2 new systems (AML [23, 24] and LogMap [25, 26]). According to matching results, there is a correlation between whether the ontologies are WIO and the average matching results of ontology matching system over them.

Therefore, it is necessary to find a new way to discover alignments for weak informative ontologies. Researchers have to utilize ontology structure information to compensate for the lack of literal information. Although an ontology can be represented as a graph, ontology matching is not equal to the graph matching problem, in which a correspondence between two elements not only means that the elements are similar in geometrical perspective, more importantly, but also are similar in semantics. Moreover, graph matching is an NP problem [14], so it cannot match ontologies efficiently. For example, we attempt to use the graph matching API provided by SOQA - SimPack [27] to match ontology graphs, and we find that it needs more than several days or even several weeks for matching two normal size ontologies. Most importantly, graph topology information cannot represent semantics in ontologies, so a geometrical graph similarity cannot imply the semantic similarity. Therefore, it is not suitable to treat ontology matching as the graph matching problem.

Most structure-based ontology matching methods are inspired by the simple idea: *similar* objects are related to similar objects. This idea also derives some heuristic rules, such as concepts may be similar when their super/sub concepts are similar and concepts may be

⁴https://wordnet.princeton.edu

⁵http://oaei.ontologymatching.org/2008/benchmarks/

| ontology | weak info | rmative elem | nents/all elements | an/N | is WIO | matching results | | | | | | |
|----------|-----------|--------------|--------------------|----------------------|---------|-------------------------|-------|------|--|--|--|--|
| ID | concepts | properties | instances | <i>w</i> /1 v | 15 1110 | (Average, Best, Worst) | | | | | | |
| 101 | 0/34 | 0/72 | 0/55 | 0.00 | No | 0.97, | 1.00, | 0.88 | | | | |
| 201 | 3/34 | 11/72 | 0/55 | 0.07 | No | 0.81, | 1.00, | 0.12 | | | | |
| 202 | 31/34 | 65/72 | 55/55 | 0.94 | Yes | 0.46, | 0.88, | 0.05 | | | | |
| 203 | 0/34 | 5/72 | 15/55 | 0.12 | No | 0.97, | 1.00, | 0.88 | | | | |
| 248 | 31/34 | 65/72 | 22/55 | 0.75 | Yes | 0.34, | 0.81, | 0.04 | | | | |
| 250 | 31/34 | 0/8 | 22/55 | 0.55 | Yes | 0.34, | 0.53, | 0.08 | | | | |
| 252 | 24/27 | 63/73 | 21/55 | 0.70 | Yes | 0.33, | 0.82, | 0.06 | | | | |
| 254 | 33/36 | 0/8 | 23/55 | 0.57 | Yes | 0.28, | 0.43, | 0.00 | | | | |
| 258 | 31/34 | 64/72 | 0/0 | 0.90 | Yes | 0.20, | 0.62, | 0.02 | | | | |
| 260 | 29/32 | 0/9 | 23/55 | 0.54 | Yes | 0.35, | 0.57, | 0.03 | | | | |

Table 1: A survey of weak informative ontologies on the sample of OAEI2008 benchmark (δ =0.25)

similar when they have similar instances. These rules have been used by some matching systems [28, 29]. However, if ontologies only have some opaque literal information, namely the δ is high, heuristic rules usually cannot work. The reason is that the similarity between elements' neighbors cannot be determined without enough clear literal information, so it causes that the similarity between elements cannot be determined too.

A reasonable solution for this problem is similarity propagation, namely, the similarity between elements can propagate to their neighbors in the graph, then similarity propagation can produce more and more similarities. After each propagation, all similarities are normalized. The propagation process is terminated until the similarities are converged. Based on such similarity propagation idea, several similarity propagation models have been proposed [18, 19, 30, 31, 17]. Among these models, similarity flooding [17] is the most classical one. The similarity flooding includes three steps: (1) constructing pairwise connectivity graph; (2) constructing induced propagation graph; (3) computing fixpoint values for matching. Similarity flooding is a versatile matching algorithm and can be implemented easily, but it is not sensitive to the initial similarity seeds. It means that different initial seeds would produce similar matching results. Similarity flooding algorithm has been used for schema matching in database and XML data [3]. However, similarity flooding is not a perfect algorithm. Melnik and his colleagues summarized six disadvantages [17], for example, having similar neighbors is the necessary precondition of this algorithm. After we try to use similarity flooding to match ontologies directly, we also find the algorithm cannot work effectively for matching WIOs. First, similarity flooding does not consider the similarities between edges, which are properties in ontologies, so the property similarities between ontologies cannot be calculated. Secondly, the maximum pairwise connectivity graph is $N_A * N_B$ (N_A and N_B are the numbers of edges in two ontologies, respectively), and it will greatly increase the time complexity for fixpoint computing and space complexity for storing the pairwise connectivity graphs. Moreover, in real-world matching tasks, the ontology graph may have thousands edges, so the corresponding pairwise connectivity graphs would become too large to be handled. For above reasons, the similarity flooding algorithm cannot be used directly for matching WIOs. This paper aims to modify the similarity flooding and proposes a new propagation model to solve the matching problem for WIOs.

3. Overview of the Methodology

Figure 2 depicts an overview of the proposed method for matching weak informative ontologies, which involves three steps: (1) building the ontology graph from the WIO, (2) extracting semantic subgraphs from the ontology graph, and (3) calculating similarity propagation to obtain similarity matrix and the alignment.

We first use the hybrid ontology graph to represent the WIO for distinguishing multiple properties between concepts, then explicitly describe the containers and collections in the ontology graph, afterwards, enrich the ontology by discovering hidden semantics, furthermore, refine the ontology graph by removing annotation and definition triples. As a result, according to the original source WIO and target WIO, we build two ontology graphs, which can clearly describe the semantic information in ontologies.

For each concept (denoted by c in Figure 2) or property (denoted by p in Figure 2) in the ontology graph, we extract the corresponding semantic subgraph, which can precisely describe the meaning of the concept or property. This step is based on the commonsense that people can understand a concept or property with limited semantic information, rather than the whole ontology. Specially, in this paper, we apply a circuit model to efficiently rank triples and then extract semantic subgraphs. More concretely, in the circuit model, the conductivity simulates the capability of conveying information, the voltage indicates the capability of preserving information, and the current denotes the semantic information flows on edges in the ontology graph.

Based on semantic subgraphs, this paper proposes a novel similarity propagation model for matching the weak informative ontologies. Considering the characteristics of ontologies, we design a strong constraint condition during similarity propagation, which not only



Figure 2: Overview of matching weak informative ontologies

avoids the performance drawbacks of similarity flood model but also can handle the correspondences between properties in ontology matching. Additionally, the propagation model employs the updating mechanism, credible seeds, penalty, termination condition, and propagation scale strategies in order to ensure a balance between matching efficiency and quality. In particular, the initial credible seeds in propagation are provided by a matcher based on semantic subgraphs, i.e. the matcher calculates the similarities between the semantic description documents, which are constructed from semantic subgraphs. Lastly, after the similarity propagation, we obtain the similarity matrix and then extract the alignment from it.

4. Ontology Graph

An ontology is composed of statements, which are triples like $\langle s, p, o \rangle$. s, p and o stand for the subject, predicate and object in a statement, respectively. There are three kinds of ontology resources: URIs resources, literals and blank nodes. In a triple, the subject can be URIs resources or blank nodes but not literals, and the predicate must be URIs resource. Let sub(O), pred(O) and obj(O) represent the sets of ontology resources for subject, predicate and object, respectively. An ontology can be directly converted into a raw ontology graph.

Definition 4 (*Raw Ontology Graph*). An ontology O can be represented by a labeled directed graph $G_r = \langle V, E, l_V, l_E \rangle$, where V and E refer to sets of vertices and edges, respectively. $V = \{x | x \in sub(O) \cup obj(O)\}, E = \{y | y \in pred(O)\}, l_V$ and l_E are functions which map vertices and edges to their labels. Two vertices and an edge strictly correspond to a triple in an ontology. G_r is called the raw ontology graph.

Figure 3 (a) shows a simple raw ontology graph for describing conference knowledge. The hollow arrows represent rdfs:subClassOf. The vertex $_a:17$ denotes a blank node.

A raw ontology graph G_r is a multigraph, in which more than one edge can exist between two vertices. A raw ontology graph and its adjacency matrix have two shortcomings. First, a property can appear at a vertex and an edge at the same time, but the adjacency matrix cannot represent this fact. Second, the adjacency matrix cannot distinguish multiple edges between two vertices in the multigraph. To deal with the first shortcoming, we record all statements about properties with extra space. The second shortcoming can be solved by the bipartite graph, which converts each triple $\langle s, p, o \rangle$ into three triples: $e_s = \langle T_i, S, s \rangle$, $e_p = \langle T_i, P, p \rangle$ and $e_o = \langle T_i, O, o \rangle$, where T_i assures the three triples can be reconverted into the original triple. As shown in Figure 3(b), multiple edges in part of Figure 3(a) are represented as the bipartite graph.

Although the bipartite graph can distinguish multiple edges between two vertices and then all ontology resources can appear at vertices, it still has disadvantages: (1) A bipartite graph size is three times larger than the original raw ontology graph. (2) The bipartite graph cannot directly describe semantic relations between elements, and that makes it difficult to analyze the ontology graph. To avoid these disadvantages, we present the hybrid ontology graph, which combines advantages of the raw ontology graph and the bipartite graph. In



Figure 3: Ontology graph processing

a hybrid ontology graph, if two vertices have only one edge, the triple can be represented in the raw ontology graph, and if there are more than one edges between two vertices, we convert some edges into a bipartite graph.

Definition 5 (Hybrid Ontology Graph). Given a raw ontology graph $G_r = \langle V, E, l_V, l_E \rangle$, let $E_p(v_m, v_n) = \{\langle v_m, p_i, v_n \rangle \in E\}$ be all edges from v_m to v_n . $|E_p(v_m, v_n)|$ is the number of edges between v_m and v_n . The hybrid ontology graph for G_r is $G_h = \langle V', E', l_V, l_E \rangle$, which can be constructed according to following rules:

- (1) If $|E_p(v_m, v_n)| = 1$, the edges and vertices are directly converted into G_h .
- (2) If $|E_p(v_m, v_n)| > 1$, then randomly convert $|E_p(v_m, v_n)| 1$ edges into a bipartite graph and add them into G_h .

A hybrid ontology graph can be implemented by an adjacency matrix. Edges using the bipartite graph style can be reconverted into the original triples. The corresponding hybrid

ontology graph of Figure 3(a) is shown in Figure 3(c), in which an edge is represented by bipartite graph.

There is some semantic information hidden in the ontology which would be useful for understanding the ontology, meanwhile, some triples are not useful for describing the semantics of elements. Therefore, a hybrid ontology graph needs to be further processed, that includes processing containers and collections, enriching and refining the ontology graph.

Phase 1: Processing containers and collections

In RDF language⁶, containers and collections are used to describe sets of resources. Containers are represented in rdf:Bag, rdf:Seq and rdf:Alt. Collections are represented in rdf:List. Although container and collection simplify the ontology, but their semantics cannot be clearly represented in ontology graphs. Hence, we use semantically equivalent statements to replace containers and collections. For example, in a triple $< Beatles, artist, _a11 >$, $_a11$ is a rdf:Bag blank node and has four members: John, Paul, George, Ringo, then we use four triples to replace the original description as: < Beatles, artist, John >, < Beatles, artist, Paul >, < Beatles, artist, George >, < Beatles, artist, Ringo >.

Phase 2: Enriching the ontology graph

The enriching process discovers more hidden semantics and represents them clearly. In fact, although discovering hidden semantics can be seen as ontology reasoning, we use enriching rules instead of existing reasoners because the rules have more flexible operations on ontologies and are independent of ontology languages.

- Step 1. Enriching domain and range: In the property hierarchy, the domain and range of a super property can be inherited by its sub properties. Such semantics are clearly declared in ontology graphs.
- Step 2. Enriching concept axioms: Concept axioms include owl:oneOf, owl:intersectionOf, owl:unionOf, owl:equivalentClass, etc. For example, if an owl:intersectionOf axiom defines a complex concept $A \sqcap B$ which has a sub concept C, then $A \sqsupset C$ and $B \sqsupset C$ are added into the ontology graph.
- Step 3. Enriching property axioms: Property axioms include owl:SymmetricProperty, owl:TransitiveProperty, owl:equivalentProperty, etc. For example, an axiom owl:SymmetricProperty declares predicate p is symmetric, and there is a triple < s, p, o > (s is subject and o is object), then a new triple < o, p, s > is added into the ontology graph.
- Step 4. Enriching *owl:sameAs* axioms: If some resources are declared by owl:sameAs, they share same semantic information.
- Step 5. Enriching properties in the concept hierarchy: The property of a super concept can be inherited by its sub concepts. For example, given < p, rdfs:domain, A > and < B, rdfs:subClassOf, A >, then < p, rdfs:domain, B > holds.

Phase 3: Refining the ontology graph

In an ontology graph, annotation and definition triples are removed, that make relations between elements clearer.

⁶https://www.w3.org/RDF/

| | G_r | phase1 | phase2 | phase3 | G_h |
|------------|--------|--------|--------|--------|-------|
| benchmark | 3772 | 3734 | 6086 | 4409 | 4453 |
| conference | 2133 | 2133 | 2774 | 2436 | 2436 |
| directory | 18970 | 18970 | 18970 | 9483 | 9483 |
| anatomy | 51268 | 51268 | 51268 | 28367 | 28367 |
| food | 337037 | 337037 | 337037 | 93312 | 93312 |
| library | 161806 | 161806 | 161806 | 20156 | 20156 |

Table 2: Changes of ontology graph size in ontology graph processing

- Step 1. Removing annotations: Remove triples including rdfs:label, rdfs:comment, rdfs:seeAlso, rdfs:isDefineBy and owl:AnnotationProperty.
- Step 2. Removing the ontology head information: The ontology head information, which is between two $\langle owl:Ontology \rangle$ tags, only describes the information about the ontology. So it can be removed.
- Step 3. Removing the version information: We remove the version information such as *owl: versionInfo, owl: backwardCompatibleWith, owl:incompatibleWith, owl:priorVersion, owl:DeprecatedClass* and *owl: DeprecatedProperty.*
- Step 4. Removing *rdf:type* statements pointing to the metadata: For example, *<Game*, *rdf:type*, *owl:Class>* should be removed, but *< StarWars*, *rdf:type*, *Game >* should be kept.
- Step 5. Removing *owl:Thing* and *owl:Nothing*: They are the global concept \top and the empty concept \bot , which do not describe the semantic relations between elements.

In Figure 3(c), red edges are added after ontology graph processing.

Table 2 shows changes of graph size in ontology graph processing. Here, we use six datasets in OAEI2007⁷: (1) 101, 301, 302, 303 and 304 in the benchmark task, (2) *Cmt* and *Edas* in the conference task, (3) *source* and *target* in the directory task, (4) *mouse* and *nci* in the anatomy task, (5) AGOVOC, NALT and GEMET in the food task and (6) Brinkman and GTT in the library task. Column 2 to 5 show changes of ontology graph size during different processing phases. When an ontology has containers and collections, phase 1 adds new triples and deletes old triples for the containers or collections, so the graph size change is uncertain. The enriching phase usually increases the graph size. Many hidden semantic information was discovered in the benchmark. The refining phase reduces the ontology graph size, especially in the last 4 datasets that have a lot of annotation information.

5. Semantic Subgraph

The semantic subgraph, which is the foundation of our solution for matching weak informative ontologies, is used for precisely describing the meaning for each ontology element. An

⁷http://oaei.ontologymatching.org/2007/

ontology is composed of elements including concepts, properties, and instances. Understanding ontology elements is one of the most frequent operations in ontology applications such as semantic searching and ontology matching. A single element is meaningless without the semantic context, which contains relative statements or triples in an ontology. The meaning of an element in an ontology can be described by relative statements or triples. However, there is no way to determine which statement should be selected and whether the selected statements are enough. For the former question, people often intuitively think statements that directly contain the element are what we need, but it is not always correct. For example, given an element *MichaelJordan*, if we know < *MichaelJordan*, *playIn*, *Bulls* >, the direct statement < *MichaelJordan*, *drive*, *car* > conveys less information about *MichaelJordan* than the indirect statement < *Bulls*, *isA*, *NBATeam* >. In order to answer the latter question, we must find a way to measure whether we have gathered enough semantic contexts. The semantic context of an ontology element is composed of some relevant statements (or triples), which is a subgraph in the ontology graph. Such subgraphs are called semantic subgraphs of ontology elements.

Definition 6 (Semantic Subgraph). Given an element e in a hybrid ontology graph G_h , its semantic subgraph $G_s(e)$ is composed of top-k (top- $k \in \mathbb{N}$) related triples that describe e. $G_s(e) \subseteq G_h$ and $G_s(e)$ has following features:

- 1. The size of $G_s(e)$ is limited. We believe that only top-k related triples can accurately describe the context of an element, namely, the semantic interpretation of an element does not need all knowledge in the ontology.
- 2. $G_s(e)$ does not emphasize semantic completeness. A semantic subgraph collect as much information about the element as possible until it can distinguish an element from other elements.
- 3. $G_s(e)$ is unique. Two elements with different semantics have different semantic subgraphs.
- 4. $G_s(e)$ prefers triples related to e. Different triples have different capabilities for describing an element. Ranking all triple about e according to related scores from high to low, $G_s(e)$ fist selects higher triples.
- 5. Closer triples do not mean more related to e. It is possible that triples far from the element may be more important than closer ones.

These five features assure that a semantic subgraph provides a clear, accurate and credible semantic description for an ontology element. Concept and property are the two important basic types of elements in an ontology. We will discuss the method for extracting semantic subgraphs for given concepts or properties from ontology graphs.

6. Extracting Semantic Subgraphs based on Circuit Model

According to the principle of semantic subgraphs, in order to extract the semantic subgraph for a given element, we first rank all related triples in ontology graphs, then select top-k triples to compose the semantic subgraph. This paper proposes a circuit model to efficiently rank triples and extract semantic subgraphs. In this section, we first introduce the circuit model, then calculate conductivity, finally discuss the extraction algorithm.

6.1. Circuit model

In order to extract semantic subgraphs, this paper utilizes circuit to model the semantic information propagation in an ontology. First, assuming that semantic information about sin an ontology graph is a measurable value in [0, 1], and it comes from vertex s with initial semantic information value 1, then flows to other vertices s_i through some triples. Since these triples have resistance to semantic information flow, the semantic information reaching s_i will have some losses. Such information about s will flow in the ontology graph continually. The semantic subgraph of s is composed of paths which start from s and have more semantic information about s. This process corresponds to an electrical circuit model. The semantic information from s corresponds to adding +1 volt on s. The semantic information on a path from s to s_i corresponds to the current reaching s_i through this path. The resistance in information flow corresponds to the electric resistance.

Similar circuit model has been used by Faloutsos et al [32] for discovering connection subgraphs in social networks. Based on this previous work, this paper proposes a modified model based on characteristics of ontologies for extracting semantic subgraphs.

In the circuit model, the capability of conveying information is the conductivity C, the capability of keeping information is the voltage V, and current I denotes the total information flows on edge per unit time.

A connection subgraph connects the source vertex s and the target vertex t [32]. However, a semantic subgraph only has the source vertex s (the given element). Therefore, the first modification is adding a sink node z as the target vertex into the ontology graph, in which each vertex has an edge to z.

Given two vertices u and v, let I(u, v) denote the current from u to v, V(u) and V(v) be the voltages on u and v, and C(u, v) and R(u, v) be the conductivity and resistance on the edge between u and v, C(u, v) = 1/R(u, v). Then an ontology graph is converted into a circuit, which has the following initial conditions:

$$V(s) = 1, V(z) = 0$$
(1)

All voltages and currents in the circuit can be computed according to Ohm's law and Kirchhoff's law.

The conductivity from any vertex to sink node z is:

$$C(u,z) = \lambda \sum_{w \neq s} C(u,w)$$
⁽²⁾

Here, λ denotes the current loss coefficient, and $1 \ge \lambda > 0$.

Definition 7 (*Delivered Current*). The delivered current $\hat{I}(P)$ in a prefix-path $P = (s = u_1, \ldots, u_i)$ is the volume of electrons that arrives at u_i through P.

The delivered current can be calculated by:

$$\hat{I}(s = u_1, ..., u_i) = \hat{I}(s = u_1, ..., u_{i-1}) \frac{I(u_{i-1}, u_i)}{I_{out}(u_{i-1})}$$
(3)

 $I_{out}(u)$ is the total current come from u.

In physics, the delivered current describes the remaining current through a path from s. Here it denotes the amount of semantic information about s in a path. In other words, the delivered current measures the relevance between a path and s, that further means that triples in the path are relevant to s. Therefore, a subgraph about s can be regarded as the combination of some prefix-paths. The captured flow of a subgraph can be defined as follows.

Definition 8 (*Captured Flow*). The captured flow of subgraph G_s is the sum of all the delivered current in the prefix-path in G_s :

$$CF(G_s) = \sum_{P=(s,\dots,t)\in G_s} \hat{I}(P)$$
(4)

Therefore, for all subgraphs having k triples, the subgraph with the maximum capture flow is the semantic subgraph. In other words, a semantic subgraph is determined by the captured flow on paths, which contain relevant triples about s. A subgraph with more captured flow has more information about s. In the above process, we do not rank the triples directly, but use the maximum captured flow to indirectly realize the ranking. Namely, a new triple is added in a semantic subgraph is always the one that can increase maximum captured flow.

Extracting semantic subgraph can be divided into two sub problems: (1) Traversing all prefix-paths from s to z and calculating their delivered currents. (2) Searching all k-size subgraphs combined by prefix-paths and calculating their capture flows. The subgraph with the maximum captured flow is the semantic subgraph. However, the two sub problems are NP problems. Faloutsos et al. proposed a greedy algorithm called *DisplayGeneration* [32] to efficiently discover connection subgraphs. The greedy idea prefers prefix-paths bringing the maximum fraction between delivery current and new nodes. Our algorithm of extracting semantic subgraphs is also based on this greedy algorithm.

The time complexity of solving the circuit model is $O(|V|^3 + |E| \times m \times k)$. The first part is the complexity of solving the circuit linear equation, and the later part is the complexity of extracting the semantic subgraphs by the greedy algorithm. |V| is the number of vertex in ontology graph, |E| is the number of edges, m is the maximum length of the paths from s to z and m < |V|. k is the size of the semantic subgraph. Usually, k is a constant and is far smaller than |V|. Most ontology graphs are sparse , in which |E| and |V| are linear relationship. Thus the later part is $O(|V|^2)$ approximately. Then the performance is dominated by the time for solving the circuit linear equation. Therefore, the approximate time complexity is $O(|V|^3)$

The space complexity of the circuit model is $O(|V|^2 + |V|m)$, where the former part is the space for solving the linear equation, and the later part is the space for the greedy algorithm. The total space complexity can also be simplified to $O(|V|^2)$.

6.2. Conductivity calculation

If the conductivity between vertices was 1, which means that the whole information is delivered. However, in an ontology graph, the semantic information would have losses when it flows through triples, so the edge conductivity should be a value in [0,1]. Based on some ontology characteristics, we first derive some heuristic rules to measure capability of delivering information, then calculate the conductivity.

Rule 1: Frequency rule

Here we first introduce an attenuation function:

$$g(x,m) = \frac{1}{2} \left(\frac{1}{x} + \left(1 - \frac{\log x}{\log(m+\varepsilon)}\right)\right), m \ge x \ge 1$$
(5)

where x and m are variables, and ε is a small positive constant to assure $log(m + \varepsilon) > 0$. Comparing with f(x) = 1/x, g(x, m) can slowly decrease when x increases.

Let f(e) be the number of triples in an ontology graph G_h in which e appears. The weight of e is:

$$\mu_f(e) = g(f(e), \max_{e_i \in G_h} f(e_i)) \tag{6}$$

The frequency rule applies to concept, property and metadata, and it means more frequently used elements deliver less information.

Rule 2: Hierarchy rule

Hierarchy is the important way to organize concepts and properties. The higher level elements have less capability of delivering information. It can be measured by following weight.

$$\mu_H(e) = \frac{dh(e)}{\max_{e_i \in G_h} dh(e_i)} \tag{7}$$

dh(e) denotes the depth of the element in the hierarchy. The hierarchy rule applies for concept and property.

Rule 3: Instance space rule

An instance space in an interpretation function on an ontology O. Given a concept C, its instance space is the set $I_{sp}(C) = \{a | < a, rdf:type, C > \in O\}$. Given a property P, its instance space is $I_{sp}(P) = \{ < a, b > | < a, P, b > \in O, a \in Dom(P), b \in Rng(P) \}$, where Dom(P) and Rng(P) are the domain and range of P, respectively.

A larger instance space indicates that a concept has more instances, then the concept has a higher possibility of being on the top level, which means that it will deliver less information. Similarly, if a property instance space is larger, then the property is used more frequently, so it conveys less information. Such a weight can be calculated as:

$$\mu_{Isp}(e) = g(|I_{sp}(e)|, max_{e_i \in G_h}|I_{sp}(e_i)|)$$
(8)

where $|I_{sp}(e)|$ is the size of the instance space of e.

Rule 4: Instance property description rule This rule is based on the assumption that more key instances would have more relative triples. This assumption is based on our experience that key instances in an ontology are described in more details than satellite

instances. Therefore, this assumption usually holds. Given an instance a and its relative triples like $\langle a, p_m, b \rangle$, the number of p_m can be used to measure the importance of the instance. The formula to calculate the weight is:

$$\mu_{Id}(a) = \frac{dp(a) + op(a)}{max_{a_i \in G_h} dp(a_i) + max_{a_i \in G_h} op(a_i)}$$
(9)

where dp(a) denotes the number of *DatatypeProperty* and op(a) denotes the number of *ObjectProperty*.

Rule 5: Few instance rule

Let a be an instance of concept C. Concepts having few instances will have more information. The weight can be calculated as:

$$\mu_{Io}(a) = g(|C(a)|, \max_{a_i \in G_h} |C(a_i)|)$$
(10)

where |C(a)| is the number of instances of C.

Based on the above five rules, the weights for concept C, property P, instance I and metadata M are as follows:

$$\mu(C) = \gamma_{C1} \times \mu_f(C) + \gamma_{C2} \times \mu_H(C) + \gamma_{C3} \times \mu_{Isp}(C)$$
(11)

$$\mu(P) = \gamma_{P1} \times \mu_f(P) + \gamma_{P2} \times \mu_H(P) + \gamma_{P3} \times \mu_{Isp}(P)$$
(12)

$$\mu(I) = \gamma_{I1} \times \mu_{Id}(I) + \gamma_{I2} \times \mu_{Io}(I)$$
(13)

$$\mu(M) = \mu_f(M) \tag{14}$$

where $\gamma_{C1} + \gamma_{C2} + \gamma_{C3} = 1$, $\gamma_{P1} + \gamma_{P2} + \gamma_{P3} = 1$, and $\gamma_{I1} + \gamma_{I2} = 1$.

Finally, the conductivity for any triple $t = \langle s, p, o \rangle$ can be obtained based on the weights of s, p and o. Since s and o are relevant to other triples, their weights should be divided by the degrees.

$$w(t) = \frac{\frac{\mu(s)}{\deg ree(s)} + \mu(p) + \frac{\mu(o)}{\deg ree(o)}}{3}$$
(15)

Although five rules are intuitive and empirical, they essentially conform to entropy in information theory, namely, lower possibility events deliver more information.

6.3. Semantic subgraph extraction algorithm

After calculating the conductivity, the circuit model can be used to extract semantic subgraphs for concepts and properties. Algorithm 1 describes the extraction process. A concept always locates at a vertex in an ontology graph. First, for any concept e_i , 1 volt is added to the vertex (line 5). Then, the circuit equation is solved (line 6). Third, according to greedy algorithm *DisplayGeneration* [32], the k-size subgraph with the maximum captured flow is the semantic subgraph (line 7). Finally, semantic subgraph $G_s(e_i)$ is obtained. Note that, since a property can appear on a vertex and an edge at the same time, the extraction for a property has little different to a concept. For a property P_i , if 1 volt was only added to P_i , the triple $\langle c_j, P_i, o_j \rangle$ would have less current, but it is useful for describing the Algorithm 1: Extracting semantic subgraphs

Input: ontology graph G Output: S:semantic subgraphs for elements

```
1 begin
       S \leftarrow \varnothing
 2
        // calculate weights
       G_w \leftarrow GetGraphTripleWeight(G)
 3
        // traverse all elements
       foreach e_i \in G do
 4
            // add 1 volt
           SetVolt(e_i) \leftarrow 1
 5
            // solve circuit equation
           SolveLinearSystem(G_w, e_i)
 6
            // extract a semantic subgraph
           G_s(e_i) \leftarrow Extract(G_w, e_i, N)
 7
           S \leftarrow S \cup G_s(e_i)
 8
       end
 9
10 end
```

semantic of P_i . Therefore, an edge from P_i to c_j is added to overcome this problem, that will increase the current on edge $\langle c_j, P_i, o_j \rangle$.

Figure 4 shows the process of extracting the semantic subgraph for concept *Paper* in Figure 3. In Figure 4(a), sink node z is added and all conductivities are calculated. In Figure 4(b), +1 voltage is added on *Paper* and z has 0 voltage, then all voltages and currents are computed. Red values are voltages and green values are currents. Figure 4(c) shows some paths from *Paper* to z and their delivered currents. All paths are ranked by the delivered currents. In order to extract a 5-size semantic subgraph for *Paper*, we can add these paths one by one according to their delivered currents from high to low, until the semantic subgraph has 5 triples. Such subgraph is the semantic subgraph of *Paper*. The 5 triples in this semantic subgraph are more relevant to *Paper* than the ones in paths with lower delivered currents. The captured flow of this semantic subgraph is maximum in all subgraphs with 5 triples.

7. Similarity Propagation Model Based on Semantic Subgraphs

Based on semantic subgraphs, we propose a novel similarity propagation model for matching weak informative ontologies. We first discuss the similarity propagation condition, then present the detail of the propagation model including the updating, seeds, penalty, and termination condition.



Figure 4: An example of extracting the semantic subgraph for a concept *Paper*

7.1. Similarity propagation condition

Ontology graph consists of triples like $\langle s_i, p_i, o_i \rangle$. In ontology matching, a reasonable similarity propagation should consider both vertices $(s_i \text{ and } o_i)$ and edges (p_i) . Similarity flooding model [17] presumes that each edge pair (p_x, p_y) has 1.0 similarity value, and the similarity of vertex pair (s_x, s_y) will be propagated to another vertex pair (o_x, o_y) . This propagation condition has three disadvantages: (1) It would produce a large number of alignment candidates and generate a large scale pairwise connectivity graph; (2) It would produce many incorrect alignment candidates. (3) It cannot deal with the correspondences between properties in ontology matching.

In order to avoid these disadvantages, this paper proposes a new propagation condition for ontology matching, namely, the strong constraint condition (SC-condition).

Definition 9 (*Strong Constraint Condition*). Given two triples $t_i = \langle s_i, p_i, o_i \rangle$ and $t_j = \langle s_j, p_j, o_j \rangle$, and let S_s , S_p and S_o denote the corresponding similarities of (s_i, s_j) , (p_i, p_j) and (o_i, o_j) , respectively. Similarities can be propagated only t_i and t_j satisfy following three conditions:

- (1) In S_s , S_p and S_o , at least two similarities must be larger than threshold θ ;
- (2) If t_i includes ontology language primitives, the corresponding positions of t_j must be same primitives;
- (3) t_i or t_j has at most one ontology language primitive.

Condition (1) ensures that the final similarity results are creditable after propagating. The ontology language primitives refer to RDF vocabularies and OWL vocabularies⁸. Condition (2) ensures that two triples use same ontology language primitive to describe semantics. For example, $<Conference_Paper, rdfs:subClassOf, Paper >$ and <Paper, rdfs:subClassOf, Document> use the RDF primitive rdfs:subClassOf as predicate, so the similarities can be propagated between them. Condition (3) ensures that there is no ontology definition and declaration triples during propagating, because such triples may cause incorrect matching results. For example, without condition (3), two triples <PhDStu, rdf:type, rdfs:Class> and <Paper, rdf:type, rdfs:Class> will cause wrong alignment: PhDStu = Paper.

After one propagation, the similarity of an element pair will be increased by the sum of other two pairs. Taking the similarity S_s as an example after i^{th} propagation, its new similarity is:

$$S_{s}^{i} = S_{s}^{i-1} + w_{po} \times S_{p}^{i-1} \times S_{o}^{i-1}$$
(16)

Analogously, the S_p^i and S_o^i are:

$$S_{p}^{i} = S_{p}^{i-1} + w_{so} \times S_{s}^{i-1} \times S_{o}^{i-1}$$
(17)

$$S_o^i = S_o^{i-1} + w_{sp} \times S_s^{i-1} \times S_p^{i-1}$$
(18)

 w_{po}, w_{so} and w_{sp} are propagation factors, which will be discussed latter. In addition, all similarities will be normalized after each propagation.

7.2. Similarity propagation model

Our new similarity propagation model contains three steps. Given two ontology graphs and initial similarity seeds, we first construct pairwise connectivity graph, then get induced propagation graph, and finally obtain the new similarities by fixpoint value calculation. Figure 5 illustrates the similarity propagation model with three steps as follows.

Step1: Constructing pairwise connectivity graph

Traditional similarity flooding model is not sensitive to initial similarity seeds, so all initial similarity values can be set to 1.0. However, in ontology matching, similarity propagation cannot use the same setting, because it will not only cause very large pairwise connectivity graph but also generate many wrong correspondences. In our view, the quality of the initial similarity seeds is very important for matching ontologies, namely, credible correspondences would generate more credible correspondences during similarity propagation. Wrong correspondences are noise in similarity propagation. Therefore, this paper tries to use few high

⁸https://www.w3.org/TR/owl-guide/



Figure 5: Similarity propagation model with SC-condition

quality correspondences as initial similarity seeds, which can be calculated by string-based matching methods or provided manually.

According to initial similarity seeds and the SC-condition, the pairwise connectivity graph can be constructed as shown in Figure 5. The pairwise connectivity graph is influenced by similarity seeds. Different seeds will cause different pairwise connectivity graphs.

Step2: Constructing induced propagation graph

According to formula (16)-(18), similarities from two element pairs are propagated to the third pair. The propagation factor measures how much similarity value can be propagated. There are three kinds of propagation factor: w_{sp} , w_{so} and w_{po} .

Take w_{sp} as an example, it denotes how much similarity value that comes from S_s and S_p can be propagated to S_o . Let f_{sp} denote the number of the triple pairs having $(s_i, s_j) \xrightarrow{(p_i, p_j)} (o_x, o_y)$ style in pairwise connectivity graph, then $w_{sp} = 1/f_{sp}$. w_{so} and w_{po} can be calculated analogously.

The induced propagation graph can be represented by a bipartite graph as shown in Figure 5, where S_1 , S_2 , S_3 and S_4 in induced propagation graph denote the four triples in pairwise connectivity graph. Therefore, the similarities of properties in ontologies can also be propagated. The weights of edges denote the propagation factors. In the implementation, for the reason that propagation factors can be directly obtained according to the pairwise connectivity graph, we record the propagation factors but need not to store the induced propagation graph.

Step3: Computing fixpoint values

The similarity propagation between ontology graphs can be computed iteratively until the final similarity matrix is converged. Under the SC-condition, the fixpoint values can be computed by formula (19), where normalization is omitted for clarity.

Actually, formula (19) is the synthesized style for formula (16)-(18). For each element pair (x, y), which would be subject pair, predicate pair or object pair, its new similarity in the $(i + 1)^{th}$ propagation contains four parts:

(1) The similarity in i^{th} propagation;

- (2) The propagation similarity when (x, y) is object pair;
- (3) The propagation similarity when (x, y) is subject pair;
- (4) The propagation similarity when (x, y) is predicate pair.

$$s^{i+1}(x,y) = s^{i}(x,y) + \sum_{\substack{ \in A \\ \in B}} s^{i}(a_{u}, b_{u}) \cdot s^{i}(p_{u}, q_{u}) \cdot w_{sp} + \sum_{\substack{ \in A \\ \in B}} s^{i}(a_{v}, b_{v}) \cdot s^{i}(p_{v}, q_{v}) \cdot w_{po} + \sum_{\substack{ \in A \\ \in B}} s^{i}(a_{t}, b_{t}) \cdot s^{i}(c_{t}, d_{t}) \cdot w_{so}$$

$$(19)$$

7.3. Incremental updating for pairwise connectivity graph

The SC-condition greatly reduces the scale of pairwise connectivity graph. After one similarity propagation, the similarity matrix will change, and new similarity values between elements are obtained. In Figure 5, the red pairs are new similarities after one propagation. Therefore, for the next propagation, we need to construct a new pairwise connectivity graph. However, constructing a pairwise connectivity graph is a time-consuming process, because we need to check all similarity values and select right triples from the ontology graph.

To reduce the constructing cost, we adopt an incremental updating way. After one similarity propagation, the new pairwise connectivity graph dose not need to be reconstructed entirely, but can be extended based on the previous one. We only update the parts in the pairwise connectivity graph whose similarities have been changed. If an element pair has been in the pairwise connectivity graph and its similarity is smaller than θ , we remove that element pair from the pairwise connectivity graph. If a new element pair has been discovered and its similarity is bigger than θ , we add that element pair into the pairwise connectivity graph. If two new triples satisfy the SC-condition, we add them to the new pairwise connectivity graph. Otherwise, we remove the triples that do not satisfy the SC-condition.

Figure 6 illustrates the incremental updating for pairwise connectivity graphs. The second pairwise connectivity graph is constructed based on the first one in Figure 5. The third connectivity graph is also constructed based on the second one. The red vertices and edges are new parts in pairwise connectivity graphs. New element pairs are also shown in red in the fixpoint values list.

7.4. Credible seeds

For initial similarity seeds, we regard correspondences having high similarity value as credible seeds. We will keep these credible correspondences during the similarity propagation. This strategy has two advantages. First, it assures that some correct correspondences



Figure 6: Updating pairwise connectivity graph

cannot be changed or be affected by other triples during similarity propagation. Second, it can avoid some unnecessary similarity propagation calculation.

If $S(a_i, b_j)$ is a credible seed, then all similarity propagation calculations like $S(a_i, b_x)$ and $S(a_y, b_j)$ can be skipped. Hence, credible seeds can not only reduce the propagation cost, but also decrease the negative effect in propagation.

7.5. Penalty in propagation

For an ideal similarity matrix, correct correspondences should have higher confidence values and incorrect correspondences should have lower confidence values. However, the real-world similarity matrix is far from perfect. Therefore, it is necessary to penalize the correspondences to improve propagation results. The penalty will make little influence for correspondences having high confidence value but reduce the potential wrong correspondences having lower confidence value.

We provide two penalty factors p_a and p_b as follows.

$$p_a = \frac{s(a_i, b_j)}{max(s_{max}(a_i, b_x), s_{max}(a_y, b_j))}$$
(20)

 $s_{max}(a_i, b_x)$ is the maximum value in *i*-th row, $s_{max}(a_y, b_j)$ is the maximum value in *j*-th column. Therefore, p_a measures the ratio of similarity value $s(a_i, b_j)$ to the maximum value in its row and column.

$$p_b = \frac{1}{1 + e^{-\alpha t}}, \ t = (\frac{N+1}{n_i + 1} / \log(N+1)), \ \alpha \ge 1$$
 (21)

N is the number of columns and rows in similarity matrix. n_i is the number of correspondences whose confidence values are larger than 0 in *i*-th column and *j*-th row. We set $\alpha = 3$ in the implementation.

After being penalized, the new similarity value is:

$$S'(a_i, b_j) = S(a_i, b_j) \cdot p_a \cdot p_b \tag{22}$$

 p_a penalizes the correspondences having low similarity values, and p_b penalizes correspondences whose column and row have too many correspondences with S(x, y) > 0.

7.6. Termination condition

Our similarity propagation model should satisfy three termination conditions: (1) The matrix norm between two sequential similarity matrices is not bigger than a given threshold. Propagation should assure that the final similarity matrix is convergent. Fortunately, Melnik and his colleagues have proved that fixpoint computing can be convergent if the pairwise connectivity graph is a strongly connected graph [17]. (2) There is no update for the pairwise connectivity graph. (3) In a similarity propagation, to avoid the matrix needs too many times propagation to be convergent, we set the maximum propagation times as 8 in the implementation.

8. Propagation Scale Choosing Strategies

To improve the efficiency of similarity propagation and the quality of matching results, we design five propagation scale strategies to study what kind of graphs should be used in the similarity propagation. These strategies are about choosing the right parts in the ontology graph for similarity propagation.



Figure 7: Five propagation scale strategies

An element e has a semantic subgraph $G_s(e)$. All semantic subgraphs of concepts are combined to a graph G_{CC} . All semantic subgraphs of properties are combined to a graph G_{CP} . G_{CC} and G_{CP} may be overlapping. Let G_C be the graph combined by all semantic subgraphs of concepts and properties. Then $G_C = G_{CC} \cup G_{CP}$. Meanwhile, G_C is part of hybrid ontology graph G_h . The left of Figure 7 illustrates the intersection relationships of these graphs. According to these graphs, we design five propagation scale strategies as shown in the right of Figure 7.

Strategy 1: Full ontology graph propagation

In similarity propagation, a simple and direct way is using full ontology graph as the propagation scale. For two ontologies O_A and O_B , the similarity propagation is between two hybrid ontology graph G_h^A and G_h^B . Although there is no ontology information to be missed in this strategy, it also has two drawbacks: (1) During similarity propagation, the pairwise connectivity graph may become too large, then it will influence the efficiency of propagation and be difficult to be handled. Especially for the large scale ontology graph, it is possible to generate very large pairwise connectivity graph. (2) More triples do not mean better propagation results. In full ontology graph, some triples are not important for describing semantics. In addition, too many triples may increase more uncertainty or noise in propagation and bring negative affection for matching results.

Strategy 2: Independent semantic subgraph propagation

A semantic subgraph is used to precisely describe an element. Therefore, if we constrain the propagation scale in semantic subgraphs, the propagation can avoid triples that is irrelevant to the element. Given two elements e_i and e_j and corresponding semantic subgraphs G_{Si}^A and G_{Sj}^B , the similarity S(a, b) is obtained by the similarity propagating between G_{Si}^A and G_{Sj}^B . If two ontologies have n and m elements respectively, this strategy needs $n \times m$ times similarity propagations. For the reason that each semantic subgraph is small, the similarity propagation can be calculated quickly.

Strategy 3. One combined semantic subgraph propagation

We notice that different semantic subgraphs may be overlapping. In strategy 2, some triples would be used in multiple similarity propagations. Therefore, we can combine all semantic subgraphs as the propagation scale. According to features of semantic subgraphs, the combined graph only contains triples relevant to elements. It not only improves the propagation efficiency, but also removes the irrelevant information to avoid introducing propagation noise. Let G_C^A and G_C^B be the combined semantic subgraphs of O_A and O_B , respectively, the similarity propagation is between G_C^A and G_C^B .

Strategy 4: Two separate combined semantic subgraphs propagation

Strategy 3 can be redivided into a more concrete strategy. Since a G_C is combined by a G_{CC} and a G_{CP} , the similarity propagation between concepts can be constrained in G_{CC} , and the similarity propagation between properties can also be constrained in G_{CP} . Therefore, for ontology O_A and O_B , the similarities between concepts are calculated by the similarity propagation between G_{CC}^A and G_{CC}^B , and the similarities between properties can be calculated by the similarity propagation between G_{CP}^A and G_{CP}^B . From the ontology matching perspective, such propagation strategy will produce two similarity matrices: one is similarity matrix for concepts, another is similarity matrix for properties.

Strategy 5: Hybrid semantic subgraph propagation

In strategy 2, the similarity propagation is between two small semantic subgraphs. In strategy 3, the similarity propagation is between two graphs combined by semantic subgraphs. To balance the above two strategies, we propose a hybrid propagation scale strategy. In this strategy, one graph is a semantic subgraph of an element e_i , and the another graph is the combined graph G_C . It means that the similarity propagation is between G_{Si}^A and G_C^B or between G_C^A and G_{Sj}^B . Therefore, after one similarity propagation, we can obtain the similarities about e_i to all elements in another ontology. This strategy only needs n times propagations.

For hybrid semantic subgraph propagation scale strategy, given an element a_i , we can get a set of similarities $\{S(a_i, b_x)\}(x = 1, ..., n)$ in the similarity propagation. Given another element b_j in the opponent ontology, we can also get another similarity set $\{S(a_y, b_j)\}(y = 1, ..., m)$. Therefore, we have two similarity matrices, in which the similarity value at $S(a_i, b_j)$ may be different. This paper calculates the average of two similarity matrices as the final propagation result. The two similarity matrices have the function of cross validation, so it can improve the quality of propagation result.

9. Matcher based on Semantic Subgraphs

Moreover, we propose a matcher based on semantic subgraphs for inputting credible aliment seeds to similarity propagation. This matcher first constructs the semantic description document for each element. Then it calculates the similarities between elements based on the semantic description documents.

9.1. Semantic description document

For an element, this paper organizes relevant literal information based on semantic subgraphs as virtual document [33]. We call this virtual document the *semantic description document* (SDD). To avoid introducing irrelevant literal information, SDD is constrained in semantic subgraphs. In addition, SDD does not consider ontology language primitive, such as *rdfs:Class* and *owl:hasValue*. In SDD construction, the text preprocessing contains stemming and removing frequent vocabularies.

For each concept, property or instance, it has a basic SDD, which consists of local name, label and annotation. The basic SDD of element e is:

$$D_{base}(e) = \varphi_1 * W_{localname} + \varphi_2 * W_{label} + \varphi_3 * W_{comment} + \varphi_4 * W_{otherAnnotation}$$
(23)

where $W_{localname}$ is local name, W_{label} is *rdfs:label* text, $W_{comment}$ is the *rdfs:comment* text, and $W_{otherAnnotation}$ is other annotation text. Weight φ_i is in [0,1]. Hence, SDD is the set of words with weights. + denotes the union operation between sets.

Since an ontology graph is enriched, elements with *owl:equivalentClass* or *owl:sameAs* axioms will have same SDD.

Generally, we consider the SDD in two sides: (1) SDD can re-organize literal information according to the semantic description of elements; (2) To avoid containing irrelevant and unimportant literal information, SDD is constrained in semantic subgraphs.

The SDD of concept C is organized by concept hierarchy, axioms, related properties and instances. Three virtual documents are constructed to describe the semantic context for sup-concepts, sub-concepts and sibling concepts. sup(C), sub(C) and sib(C) are sets of super-concepts, sub-concepts and sibling concepts of C, respectively. $dist(C, C_i)$ is the distance between C and C_i .

$$D_{\sup}(C) = \sum_{C_i \in \sup(C)} \frac{1}{dist(C, C_i)} D_{base}(C_i)$$
(24)

$$D_{sub}(C) = \sum_{C_i \in sub(C)} \frac{1}{dist(C_i)} D_{base}(C_i)$$
(25)

$$D_{sib}(C) = \sum_{C_i \in sib(C)} D_{base}(C_i)$$
(26)

Virtual documents of related properties of concept C are follows.

$$D_{dom}(C) = \sum_{P_i \in Dom^*(C)} D_{base}(P_i)$$
(27)

$$D_{rng}(C) = \sum_{P_i \in Rng^*(C)} D_{base}(P_i)$$
(28)

where $Dom^*(C)$ and $Rng^*(C)$ are properties whose domain and range are C.

A virtual document is used to describe direct instances of C:

$$D_{ins}(C) = \sum_{c_i \in C} D_{base}(c_i) \tag{29}$$

The SDD of property P is organized by domain and range statements, and it contains two parts:

$$D_{dom}(P) = \sum_{C_i \in Dom(P)} D_{base}(C_i)$$
(30)

$$D_{rng}(P) = \sum_{e_i \in Rng(P)} D_{base}(e_i)$$
(31)

An ontology often has blank nodes, which are contained by SDD of concepts and properties. The SDD of a blank node b is as follows.

$$D_{blank}(b) = \alpha_1 * \sum_{t_i \in C1} D_{base}(pre(t_i)) + D_{base}(obj(t_i)) + \alpha_2 * \sum_{t_l \in C2} D_{base}(sub(t_l)) + D_{base}(pre(t_l)) + \alpha_3 * \sum_{t_m \in C3} D_{base}(pre(t_m)) + D_{blank}(obj(t_m)) + \alpha_4 * \sum_{t_n \in C4} D_{blank}(sub(t_n)) + D_{base}(pre(t_n))$$
(32)

Let B be the set of blank nodes in ontology O. Given a triple $t = \langle s, p, o \rangle$, sub(t), pre(t) and obj(t) are subject, predicate and object. C1, C2, C3 and C4 are sets of triples:

$$C1 = \{t \in O | sub(t) = b \text{ and } obj(t) \notin B\}$$

$$C2 = \{t \in O | sub(t) \notin B \text{ and } obj(t) = b\}$$

$$C3 = \{t \in O | sub(t) = b \text{ and } obj(t) \in B\}$$

$$C4 = \{t \in O | sub(t) \in B \text{ and } obj(t) = b\}$$

 α_i is a weight. Let dist(t, b) be the distance between t and b, then $\alpha_i = 1/dist(t, b)$. $D_{blank}(b)$ can be computed recursively. If there is a circle during recursive computing, the computing is terminated directly.

9.2. Similarity computation

After constructing SDD for concepts and properties, correspondences can be discovered by computing similarities between SDD. A SDD is a set of vocabularies with weights, namely, $SDD = \{p_1 * W_1, p_2 * W_2, ..., p_x * W_x\}$. We can use cosine to measure the similarities.

Let $Doc = \{SDD_1, SDD_2, \ldots, SDD_N\}$, and each SDD contains n items t_1, t_2, \ldots, t_n . Thus each document SDD_i can be described as an n-dimension vector $\vec{D}_i = (d_{i1}, d_{i2}, \ldots, d_{in})$, where d_{ij} is the weight of j-th item. If the edit-distance similarity of two items is larger than a predefined threshold 0.85, they are treated as same item. The weight d_{ij} in vector \vec{D}_i is TF-IDF weight.

The similarity between two virtual documents is the cosine value of vectors. Therefore, the similarity between \vec{D}_i and \vec{D}_j is:

$$Sim(\vec{D}_{i}, \vec{D}_{j}) = \frac{\sum_{k=1}^{n} d_{ik} \times d_{jk}}{\sqrt{\sum_{k=1}^{n} d_{ik}^{2} \times \sum_{k=1}^{n} d_{jk}^{2}}}$$
(33)

In addition, for the reason that we divide and organize all literal information according to semantics, this matcher based on semantic subgraphs also performs well for informative ontologies.

10. Experimental Evaluation

We have implemented the method for matching weak informative ontologies in our ontology matching system Lily ⁹. Lily is implemented in Java and C++. In this section, we first present the dataset, criteria, and settings used in the experiments. Secondly, we verify the proposed method and compare with other works. Thirdly, we discuss the effect of propagation scale strategies. Then we address the influence of initial similarity seeds and propagation performance. Finally, we discuss the results on general ontology matching tasks.

⁹https://github.com/npubird/LilyWIO

10.1. Dataset, criteria and settings

In the evaluation, we use OAEI benchmark as the dataset. The reason is that the OAEI benchmark includes not only informative ontologies but also some WIOs, which are very similar to the WIOs used in practical applications. From 2004 to 2016, even though OAEI benchmark has some changes in each year, it has little effect on the fairness of evaluation. This paper uses benchmark2008 and benchmark2009¹⁰, which are two similar versions of the OAEI benchmark. The dataset has 110 matching tasks including 50 basic matching tasks and 60 transformations from basic tasks. All matching tasks have non-sequential number from 101 to 304.

According to characteristics of the dataset, we divide it into five groups:

- 101-104: This group contains same, irrelevant, language generalized and restricted ontologies.
- 201-210: In this group, ontology structure is preserved, but labels and identifiers are replaced by random names, misspellings, synonyms and foreign names. The comments have been suppressed in some cases. These ontologies are similar to WIOs in industrial applications.
- 221-247: This group is divided into two subgroups: 221-231 and 232-247. The first subgroup contains 11 kinds of modifications. For example, the hierarchy is flattened or expanded, and individuals, restrictions and data types are suppressed. In the second subgroup, the modifications are the combinations of the ones used in 221-231.
- 248-266: Most ontologies in this group are weak informative ontologies. All labels and identifiers are replaced by random names, and comments are also suppressed. Therefore, all ontologies in this group are WIOs.
- 301-304: This group contains 4 real-world matching tasks.

We inspect the dataset manually and select the 78 WIOs as shown in Table 3. There is no WIOs in three groups: 101-103, 221-247 and 301-304. All 67 ontologies in group 248-266 are WIOs. There are 11 WIOs in group 201-210. Therefore, the ratio of WIOs in this datasets is 78/110 = 70.9%. These WIOs will be used to verify our method.

This paper uses the classical criteria: precision, recall and F1-measure to evaluate the matching results. Let Q be the alignment of our method and T be the reference alignment, then the precision, recall and F1-Measure are:

$$P = \frac{|Q \cap T|}{|Q|} \tag{34}$$

$$R = \frac{|Q \cap T|}{|T|} \tag{35}$$

F1-measure =
$$\frac{2PR}{P+R}$$
 (36)

¹⁰http://oaei.ontologymatching.org/2009/benchmarks/

| group | number of ontologies | weak informative ontologies |
|-----------|----------------------|---|
| 101-104 | 3 | N/A |
| 201 - 210 | 18 | 201-6,201-8,202,202-4,202-6,202-8,205,206,207,209,210 |
| 221 - 247 | 18 | N/A |
| 248-266 | 67 | all ontologies |
| 301 - 304 | 4 | N/A |

Table 3: Overview of OAEI benchmark(2008,2009) dataset

In the evaluation, we set $\lambda = 0.85$ in circuit model, $\theta = 0.005$ in SC-condition, and $\phi = \delta = 0.25$ for checking weak informative ontologies.

10.2. Over performance on weak informative ontologies

We verify our method on the 78 weak informative ontology matching tasks listed in Table 3. In order to simplify results, we do not list the matching result for each task, but use the prefix numbers to divide the 78 matching tasks into 22 groups. For each group, we present the average matching results. For example, the 202 group contains 4 matching tasks with prefix 202, namely, 202, 202-4, 202-6 and 202-8, and we calculate the average precision, recall and F1-measure on the 4 matching tasks. In addition, in the similarity propagation model, we use the hybrid semantic subgraph propagation strategy.



Figure 8: Matching results on weak informative ontologies

Figure 8 presents the matching results obtained by our method on the 22 groups. For group 201-210, our method produces high quality results, whose precisions are larger than 0.99 and recalls are larger than 0.87. The results mean that if the ontology structure is preserved, misspellings, synonyms, foreign names, and even labels and identifiers are replaced

by random names, our method still works perfectly. The explanation for this fact is that our method can utilize limited literal information to generate few credible seeds and then find more correct alignments by the similarity propagation model. Although all matching tasks in group 248-266 are difficult, our method also perform well on them. For most tasks in group 248-266, the precisions are larger than 0.8 and the recalls are larger than 0.7. For group 254, 262, 265 and 266, we cannot obtain results with high recalls. The main reason is that the ontology structure is not preserved, especially, the concept hierarchy is flattened and there is no property. Moreover, comments are removed and labels are scrambled by random names. Therefore, the similarity propagation cannot work without structure information, and there is, theoretically, no matching method can deal with this situation.

As shown in Figure 9, we compare the results with similarity propagation and the results without similarity propagation. The dataset used here is benchmark2008, which includes both informative ontologies and weak informative ontologies. We run our matcher based on semantic subgraphs (in Section 9) and the similarity propagation method (in Section 7 and 8), respectively. According to Figure 9, we observe the following facts.

- The similarity propagation method proposed in this paper improves the quality of matching results, especially for the weak informative ontologies, such as the ontologies in group 248-266.
- For the weak informative ontologies, our similarity propagation method can increase the recall of results greatly. The reason is that our similarity propagation model can discover more correct correspondences with few correspondences as seeds. Especially, these new correspondences are difficult to be discovered by the methods without similarity propagation. Therefore, our similarity propagation model can improve the recall of matching results.
- For informative ontologies, such as group 221-247, our method can also produces good results. It means that our method is a general matching method.

Table 4 and Table 5 compare the results obtained by different matching systems on the 22 group of weak informative ontologies in benchmark2008 and benchmark2009, respectively. Lily is our matching system with the method for matching WIOs. The first column lists the 22 group matching tasks. From the second column, each column is the F1-measure results of one system. The last row is the average F1-measure of each system. Especially, we compare similarity flooding (SF) algorithm with other ontology matching systems.

Table 4 shows the results of 17 systems, in which the results of the first 13 systems are retrieved from http://oaei.ontologymatching.org/2008/results, and results of AML¹¹ and LogMap ¹² are obtained by running original source codes, respectively. SF stands for the results of the similarity flooding algorithm. Not all systems perform well on the weak

¹¹https://github.com/AgreementMakerLight/AML-Project

 $^{^{12} \}rm https://github.com/ernestojimenezruiz/logmap-matcher$



Figure 9: Matching results without similarity propagation VS with similarity propagation (P1, R1 and F1 are the precision, recall and F1-measure for matcher based on semantic subgraphs; P2, R2 and F2 are the precision, recall and F1-measure for similarity propagation model)

informative ontologies matching tasks. In fact, 11 in 17 systems are smaller than 0.6 F1measure, and 5 systems are in 0.6 to 0.8 F1-measure. Only Lily obtains the highest 0.82 F1-measure. Specially, Lily obtains the best matching results for most matching tasks.

Besides Lily, some systems in Table 4 (such as ASMOV, aflood, RIMOM, SAMBO, GeRoMe) have their similarity propagation methods. They also have good performance on the weak informative ontologies. It demonstrates that similarity propagation is an efficient way to match weak informative ontologies.

Table 5 shows the matching results of 16 systems on weak informative matching tasks, and the first 15 results are obtained from http://oaei.ontologymatching.org/2009/results. Results of AML and LogMap is obtained by running source codes. It also can be seen that Lily is the best system. Specifically, the F1-measure of 9 systems is small than 0.6, and 6 systems have the F1-measure in 0.6 to 0.8. Only Lily and ASMOV get 0.82 and 0.8 F1-measure, respectively. For most matching tasks, Lily also obtains the best F1-measure.

It should be noted that the similarity flooding has very poor performance on the datasets. The results support our analysis that similarity flooding cannot directly deal with ontology matching.

Finally, we study some alignment cases in Table 4. First, we randomly select 4 alignments obtained by Lily in 210 task, which is a cross-lingual ontology matching task. The selected alignments are: $MastersThesis = M\acute{e}moireDeMastere, PhdThesis = M\acute{e}moireDeDoctorat, numberOrVolume = numéroOuVolume, LectureNotes = Polycopié. It is surprising that Lily discovers these alignments without any external cross-lingual lexicon or resource. Lily only uses similarity propagation with few seeds to find these correct alignments, and it performs better than some systems with string-based matchers. Then, we randomly select 4 alignments obtained by Lily in 248 task, in which most literal information is replaced by meaningless strings. The selected alignments are: numberOrVolume = dzezd, LectureNotes$

| edna aflood arrowa ASMOV CIDER DSSin GeRoMe TaxoMap MapPSO RiMOM SAMBO SPIDER AML LogMap SF Lily 201 0.32 0.83 0.99 1.00 0.85 0.95 0.86 0.27 0.41 1.00 0.65 0.65 0.85 0.52 0.48 0.32 1.00 202 0.32 0.44 0.74 0.90 0.46 0.62 0.55 0.16 0.03 0.43 0.43 0.46 0.88 0.01 0.99 205 0.34 0.69 0.99 0.99 0.76 0.92 0.85 0.00 0.31 0.99 0.61 0.71 0.76 0.12 0.40 0.16 0.99 206 0.51 0.90 0.99 0.74 0.33 0.65 0.66 0.23 0.86 0.40 0.71 0.74 0.12 0.40 0.76 210 0.52 0.77 0 |
|--|
| $\begin{array}{c ccccccccccccccccccccccccccccccccccc$ |
| $ \begin{array}{cccccccccccccccccccccccccccccccccccc$ |
| $ \begin{array}{cccccccccccccccccccccccccccccccccccc$ |
| $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$ |
| $ \begin{array}{cccccccccccccccccccccccccccccccccccc$ |
| $ \begin{array}{cccccccccccccccccccccccccccccccccccc$ |
| $ \begin{array}{cccccccccccccccccccccccccccccccccccc$ |
| $\begin{array}{cccccccccccccccccccccccccccccccccccc$ |
| 249 0.40 0.76 0.54 0.87 0.55 0.51 0.54 0.19 0.38 0.86 0.52 0.52 0.55 0.50 0.47 0.50 0.89 250 0.23 0.74 0.79 0.87 0.60 0.70 0.66 0.42 0.42 0.91 0.52 0.52 0.60 0.00 0.48 0.00 0.93 251 0.40 0.73 0.76 0.88 0.54 0.68 0.55 0.17 0.37 0.83 0.52 0.52 0.60 0.00 0.48 0.00 0.93 251 0.40 0.73 0.76 0.88 0.54 0.69 0.41 0.57 0.57 0.52 0.52 0.54 0.50 0.00 0. |
| $ \begin{array}{cccccccccccccccccccccccccccccccccccc$ |
| $\begin{array}{cccccccccccccccccccccccccccccccccccc$ |
| $\begin{array}{cccccccccccccccccccccccccccccccccccc$ |
| 253 0.41 0.60 0.49 0.84 0.52 0.51 0.47 0.19 0.38 0.82 0.52 0.52 0.52 0.49 0.46 0.49 0.87 254 0.23 0.70 0.63 0.70 0.52 0.70 0.09 0.43 0.40 0.70 0.52 0.52 0.52 0.52 0.46 0.37 0.70 |
| 254 0.23 0.70 0.63 0.70 0.52 0.70 0.09 0.43 0.40 0.70 0.52 0.52 0.52 0.52 0.46 0.37 0.70 |
| |
| 257 0.23 0.55 0.60 0.68 0.60 0.51 0.66 0.42 0.41 0.75 0.52 0.52 0.60 0.54 0.48 0.38 0.73 0.54 0.48 0.38 0.73 0.54 0.48 0.38 0.73 0.54 0.55 |
| 258 0.40 0.73 0.51 0.80 0.54 0.52 0.54 0.17 0.38 0.68 0.52 0.52 0.54 0.50 0.45 0.25 0.86 0.51 0.51 0.51 0.51 0.51 0.51 0.51 0.52 0.52 0.54 0.51 0.51 0.51 0.51 0.51 0.51 0.52 0.54 0.51 0 |
| 259 0.63 0.71 0.69 0.89 0.71 0.72 0.59 0.28 0.56 0.85 0.70 0.70 0.71 0.69 0.68 0.38 0.93 |
| 260 0.20 0.75 0.73 0.81 0.56 0.71 0.50 0.42 0.40 0.85 0.51 0.51 0.56 0.52 0.46 0.41 0.85 |
| 261 0.32 0.80 0.77 0.81 0.71 0.76 0.61 0.58 0.45 0.76 0.68 0.68 0.71 0.72 0.65 0.40 0.83 |
| 262 0.23 0.52 0.45 0.51 0.52 0.51 0.09 0.43 0.41 0.52 0.52 0.52 0.52 0.52 0.52 0.54 0.35 0.55 |
| 265 0.03 0.12 0.00 0.12 0.00 0.00 0.19 0.00 0.03 0.18 0.00 0.00 0.00 0.00 0.00 0.12 0.24 |
| 266 0.02 0.05 0.00 0.10 0.00 0.00 0.08 0.00 0.02 0.00 0.00 0.0 |
| Avg. 0.35 0.67 0.67 0.79 0.55 0.62 0.54 0.22 0.35 0.78 0.50 0.52 0.55 0.42 0.31 0.26 0.82 |

Table 4: Comparison of 17 systems on weak informative ontologies in benchmark2008 (F1-measure)

Table 5: Comparison of 16 systems on weak informative ontologies in benchmark2009 (F1-measure)

| | edna | aflood | AgrMaker | aroma | ASMOV | DSSim | ${\rm GeRoMe}$ | kosimap | MapPSO | RiMOM | SOBOM | TaxoMap | AML | LogMap | \mathbf{SF} | Lily |
|------|------|--------|----------|-------|-------|------------------------|----------------|---------|--------|-------|-------|---------|------|--------|---------------|------|
| 201 | 0.32 | 0.94 | 0.94 | 0.99 | 1.00 | 0.89 | 0.97 | 0.67 | 1.00 | 1.00 | 0.53 | 0.26 | 0.52 | 0.48 | 0.34 | 1.00 |
| 202 | 0.31 | 0.92 | 0.43 | 0.72 | 0.90 | 0.62 | 0.67 | 0.45 | 0.72 | 0.94 | 0.25 | 0.19 | 0.42 | 0.00 | 0.25 | 0.92 |
| 205 | 0.34 | 0.82 | 0.98 | 0.99 | 0.99 | 0.86 | 0.98 | 0.69 | 0.99 | 0.99 | 0.45 | 0.15 | 0.41 | 0.38 | 0.25 | 0.99 |
| 206 | 0.52 | 0.95 | 0.93 | 0.99 | 0.99 | 0.77 | 0.95 | 0.77 | 0.99 | 0.99 | 0.00 | 0.10 | 0.00 | 0.40 | 0.16 | 0.99 |
| 207 | 0.52 | 0.95 | 0.93 | 0.99 | 0.99 | 0.77 | 0.95 | 0.77 | 0.99 | 0.99 | 0.00 | 0.10 | 0.12 | 0.40 | 0.20 | 0.99 |
| 209 | 0.35 | 0.80 | 0.35 | 0.73 | 0.90 | 0.53 | 0.70 | 0.48 | 0.67 | 0.88 | 0.17 | 0.09 | 0.36 | 0.00 | 0.16 | 0.93 |
| 210 | 0.52 | 0.94 | 0.33 | 0.77 | 0.97 | 0.53 | 0.73 | 0.70 | 0.69 | 0.87 | 0.00 | 0.00 | 0.36 | 0.00 | 0.18 | 0.94 |
| 248 | 0.41 | 0.82 | 0.52 | 0.71 | 0.90 | 0.68 | 0.71 | 0.52 | 0.36 | 0.87 | 0.32 | 0.22 | 0.32 | 0.00 | 0.24 | 0.94 |
| 249 | 0.41 | 0.91 | 0.52 | 0.51 | 0.84 | 0.52 | 0.59 | 0.53 | 0.07 | 0.84 | 0.33 | 0.24 | 0.50 | 0.47 | 0.26 | 0.90 |
| 250 | 0.24 | 1.00 | 0.52 | 0.77 | 0.88 | 0.70 | 0.77 | 0.54 | 0.64 | 0.91 | 0.52 | 0.51 | 0.07 | 0.48 | 0.40 | 0.91 |
| 251 | 0.40 | 0.78 | 0.52 | 0.74 | 0.92 | 0.69 | 0.71 | 0.50 | 0.66 | 0.87 | 0.31 | 0.22 | 0.50 | 0.00 | 0.28 | 0.91 |
| 252 | 0.62 | 0.84 | 0.71 | 0.80 | 0.94 | 0.80 | 0.84 | 0.69 | 0.72 | 0.86 | 0.49 | 0.34 | 0.69 | 0.00 | 0.30 | 0.94 |
| 253 | 0.40 | 0.69 | 0.52 | 0.48 | 0.82 | 0.52 | 0.58 | 0.52 | 0.08 | 0.75 | 0.32 | 0.22 | 0.50 | 0.46 | 0.22 | 0.88 |
| 254 | 0.22 | 0.70 | 0.52 | 0.60 | 0.70 | 0.70 | 0.66 | 0.49 | 0.36 | 0.70 | 0.52 | 0.48 | 0.52 | 0.46 | 0.37 | 0.70 |
| 257 | 0.23 | 0.96 | 0.52 | 0.54 | 0.68 | 0.51 | 0.61 | 0.54 | 0.64 | 0.72 | 0.52 | 0.51 | 0.54 | 0.48 | 0.38 | 0.75 |
| 258 | 0.40 | 0.69 | 0.52 | 0.49 | 0.84 | 0.52 | 0.58 | 0.50 | 0.15 | 0.68 | 0.31 | 0.22 | 0.50 | 0.45 | 0.28 | 0.86 |
| 259 | 0.62 | 0.78 | 0.71 | 0.69 | 0.90 | 0.73 | 0.74 | 0.69 | 0.19 | 0.67 | 0.49 | 0.34 | 0.69 | 0.68 | 0.32 | 0.92 |
| 260 | 0.19 | 0.78 | 0.51 | 0.68 | 0.81 | 0.71 | 0.73 | 0.53 | 0.45 | 0.79 | 0.52 | 0.51 | 0.52 | 0.46 | 0.45 | 0.85 |
| 261 | 0.31 | 0.84 | 0.69 | 0.71 | 0.80 | 0.76 | 0.78 | 0.70 | 0.45 | 0.75 | 0.71 | 0.69 | 0.72 | 0.65 | 0.42 | 0.80 |
| 262 | 0.23 | 0.52 | 0.52 | 0.44 | 0.52 | 0.51 | 0.51 | 0.49 | 0.42 | 0.51 | 0.52 | 0.48 | 0.52 | 0.46 | 0.34 | 0.51 |
| 265 | 0.02 | 0.24 | 0.00 | 0.00 | 0.12 | 0.00 | 0.00 | 0.13 | 0.10 | 0.15 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.24 |
| 266 | 0.02 | 0.11 | 0.00 | 0.00 | 0.10 | 0.00 | 0.00 | 0.11 | 0.06 | 0.09 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.15 |
| Avg. | 0.35 | 0.77 | 0.55 | 0.65 | 0.80 | 0.61 | 0.67 | 0.54 | 0.52 | 0.76 | 0.33 | 0.27 | 0.40 | 0.31 | 0.26 | 0.82 |

= scds, proceedings = zassdzadb, institution = hsgiuyza. These results are also existing. Here, the initial seeds contain few alignments such as Address = Address and lastName = lastName, then more and more correct alignments are got by similarity propagation. The examination demonstrates that similarity propagation is a very effective way to deal with weak informative ontologies. In the inverse perspective, we find some alignments obtained by string-based systems, but the alignments are covered by results of Lily. The reason is that the matcher based on semantic subgraphs in Lily is also an effective string-based matcher.

| | 1 | | 1 1 0 | | 0 | (|
|------|------|------|-------|------|------|------|
| Size | Seed | S1 | S2 | S3 | S4 | S5 |
| 0 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 |
| 1 | 0.37 | 0.60 | 0.60 | 0.42 | 0.25 | 0.50 |
| 2 | 0.43 | 0.65 | 0.60 | 0.55 | 0.35 | 0.55 |
| 3 | 0.42 | 0.53 | 0.48 | 0.54 | 0.40 | 0.74 |
| 5 | 0.49 | 0.61 | 0.53 | 0.66 | 0.61 | 0.80 |
| 10 | 0.54 | 0.59 | 0.59 | 0.66 | 0.69 | 0.83 |
| 15 | 0.59 | 0.64 | 0.59 | 0.66 | 0.66 | 0.84 |
| 20 | 0.56 | 0.61 | 0.63 | 0.67 | 0.68 | 0.79 |
| 25 | 0.56 | 0.63 | 0.60 | 0.66 | 0.73 | 0.83 |
| 30 | 0.56 | 0.65 | 0.69 | 0.66 | 0.71 | 0.88 |
| 35 | 0.56 | 0.62 | 0.65 | 0.62 | 0.65 | 0.83 |
| Avg. | 0.53 | 0.62 | 0.61 | 0.66 | 0.68 | 0.83 |
| | | | | | | |

Table 6: Comparison of different propagation scale strategies (F1-measure)

Lily uses this matcher to provide initial seeds in similarity propagation.

Therefore, the above experimental results demonstrate that our method for matching weak informative ontologies is very effective, and the matching system with our method can perform better than other systems.

10.3. Evaluating propagation scale strategies

This experiment aims to evaluate the performance of different propagation scale strategies. The results are obtained on the 248 matching task, which is selected randomly. Similar experimental results and conclusions can also be obtained on any other weak informative matching tasks. In this experiment, we use S1, S2, S3, S4 and S5 to represent five kinds of propagation scale strategies mentioned in Section 8.

In order to demonstrate the influence of sizes of semantic subgraphs, we change sizes of semantic subgraphs from 0 to 35. A 0-size semantic subgraph only contains an element, that means the matching method only uses the information of the element. Introducing the change of the semantic subgraph size will help us to know how to set suitable size for semantic subgraphs in the similarity propagation.

Table 6 shows the experimental results (F1-measure) for comparing different strategies. Size column is the semantic subgraph size. Seed denotes the F1-measure of initial similarity seeds obtained by the matcher based on semantic subgraphs (in Section 9). Other columns are the results obtained by different propagation scale strategies. The last row provides the average F1-measure for semantic subgraphs from size 0 to 35.

According to the F1-measure of seeds, Table 6 shows that the similarity propagation can improve the matching results. The performances of similarity propagation scale strategies can be ranked as: S5 > S4 > S3 > S1 > S2. Further analyzing the experimental results, some interesting facts are observed as follows:

- Even the full graph strategy improves the quality of matching results, it is not the best propagation scale strategy.
- It is surprising that the independent semantic subgraph strategy produces the worst matching results. In this strategy, we find that some incorrect correspondences may have high similarity values during similarity propagation. Since the independent semantic subgraphs are too small, the noise similarity values cannot be suppressed in propagation.
- Combined semantic subgraph strategy (S3 and S4) can produce better results than above two strategies. That may be explained by the fact that the two strategies not only remove some irrelevant information from ontologies, but also assure that some noise similarity values can be suppressed during propagation. S3 and S4 have very close average F1-measure. Therefore, two combined semantic subgraphs strategies can improve the quality of results.
- The hybrid semantic subgraph propagation strategy (S5) is the best strategy and produces the best results. The reason is that this strategy combines advantages of S2, S3 and S4. Moreover, the cross validation function in S5 strategy also plays positive role.
- Finally, we also find that bigger semantic subgraphs cannot always produce better matching results. Therefore, we set reasonable size for semantic subgraphs in the range 10 to 15.

10.4. Influence of initial similarity seeds

The traditional similarity flooding model claims that initial similarity seeds would not affect propagation results. However, in our new model, we need to validate whether and how our similarity propagation method is sensitive to initial similarity seeds. In this experiment, we randomly modify seeds to satisfy the condition: Precision = Recall = F1-measure. The experimental data is also obtained on 248 matching task. We let F1-measure of seeds increases from 0 to 1.0 with step 0.1, then observe changes of F1-measure with different propagation strategies.

Figure 10 shows the influence of initial similarity seeds to matching results. Line B denotes the F1-measure of seeds. S1, S2, S3 and S5 are F1-measure lines for corresponding propagation scale strategies. Since S4 has similar result to S3, we omit line S4.

We observe some interesting facts in Figure 10:

- The initial seeds influence the matching result greatly. With the changes of seed F1-measure, matching results change monotonously.
- After similarity propagating, matching result is better than the initial seed. Meanwhile, better seeds would produce better matching results.



Figure 10: Influence of initial seed to matching results

• Compared to other propagation scale strategies, S5 strategy is influenced by the seed slightly. It means that S5 strategy can use seeds with low F1-measure to produce higher quality results than other strategies. Therefore, S5 is the preferred propagation scale strategy.

10.5. Propagation performance

In the practical ontology matching tasks, we find that the similarity propagation would take up 30%-50% running time. The reason is that large semantic subgraphs would cause large pairwise connectivity graphs, and the iteration process for calculating fixpoint would also terminate slowly. Therefore, the semantic subgraph size is the key factor of performance. We run our matching method with different propagation scale strategies on 248 task, then record the running time. Meanwhile, for each propagation scale strategy, the semantic subgraph size increases from 0 to 60 by 5 steps.

Figure 11 demonstrates the running time on various semantic subgraph sizes and different propagation scale strategies. We have some interesting facts as follows:

• For S1, S2 and S3 strategies, the running time has no relevant to the semantic subgraph sizes. When we change the semantic subgraph size, the lines of S1, S2 and S3 are almost



Figure 11: Performances of different propagation scale strategies

steady. This result can be explained by the fact that the size of pairwise connectivity graph will keep stable when the semantic subgraph size is larger than 5. Therefore, we can infer that the running time of S1, S2 and S3 strategies have little correlation with the semantic subgraph size.

• For the hybrid propagation scale strategy S5, Figure 11 shows that the running time quickly increases with the semantic subgraph size increasing. There are two graphs in this strategy, one is the independent semantic subgraph for an element, and another is the combined semantic subgraph. When the semantic subgraph size increases, although the size of combined semantic subgraph keeps steady, the independent semantic subgraph will grow linearly. Therefore, the pairwise connectivity graph will grow too. Finally, it needs more time to run the similarity propagation. This fact also means that we should set suitable semantic subgraph size for hybrid propagation scale strategy.

10.6. Performance on general ontology matching tasks

Our matching method for WIOs has been implemented in our ontology matching system Lily. Lily first checks whether an ontology is weak informative, then uses similarity propagation model to match weak informative ontologies and uses string-based matcher based on semantic graphs (in Section 9) to match other ontologies.

| | Tab | le 7: Ma | atching res | sults on bei | nchmark | 2008 | |
|----------------------------|--------------|---------------------|-------------|-------------------------|----------------|----------------|--------------|
| | Precision | Recall | F1-Measure | | Precision | Recall | F1-Measure |
| 101 | 1.00 | 1.00 | 1.00 | 251* | 0.96 | 0.76 | 0.85 |
| 103 | 1.00 | 1.00 | 1.00 | $251 - 2^*$ | 0.99 | 0.96 | 0.97 |
| 104 | 1.00 | 1.00 | 1.00 | 251-4* | 0.99 | 0.90 | 0.94 |
| 201 | 1.00 | 1.00 | 1.00 | $^{-}$ 251-6* | 0.94 | 0.83 | 0.88 |
| 201-2 | 1.00 | 1.00 | 1.00 | $251 - 8^*$ | 0.99 | 0.85 | 0.91 |
| 201-4 | 1.00 | 1.00 | 1.00 | 252^{*} | 0.96 | 0.79 | 0.87 |
| 201-6* | 1.00 | 1.00 | 1.00 | 252 - 2* | 0.97 | 0.94 | 0.95 |
| $201 \ 0$ $201 \ 8^{*}$ | 1.00 | 1.00 | 1.00 | 252_{-1*} | 0.07 | 0.01 | 0.95 |
| 201-0 | 1.00 | 0.84 | 0.01 | 252-4 | 0.97 | 0.04 | 0.55 |
| 202 | 1.00 | 0.04 | 0.91 | 252-0 | 0.97 | 0.94 | 0.95 |
| 202-2 | 1.00 | 0.97 | 0.90 | 202-0 | 0.97 | 0.94 | 0.95 |
| $202-4^{\circ}$ | 1.00 | 0.92 | 0.90 | 205 | 0.01 | 0.09 | 0.08 |
| 202-6* | 0.98 | 0.87 | 0.92 | 253-2* | 0.98 | 0.93 | 0.95 |
| 202-8* | 0.98 | 0.85 | 0.91 | 253-4* | 1.00 | 0.92 | 0.96 |
| 203 | 1.00 | 1.00 | 1.00 | $253-6^{*}$ | 0.95 | 0.81 | 0.87 |
| 204 | 1.00 | 1.00 | 1.00 | $253-8^*$ | 0.95 | 0.79 | 0.86 |
| 205^{*} | 1.00 | 0.99 | 0.99 | 254* | 1.00 | 0.27 | 0.43 |
| 206^{*} | 1.00 | 0.99 | 0.99 | 254-2* | 1.00 | 0.82 | 0.90 |
| 207^{*} | 1.00 | 0.99 | 0.99 | 254-4* | 1.00 | 0.70 | 0.82 |
| 208 | 1.00 | 0.99 | 0.99 | 254-6* | 1.00 | 0.61 | 0.76 |
| 209^{*} | 0.97 | 0.88 | 0.92 | 254-8* | 1.00 | 0.42 | 0.59 |
| 210^{*} | 1.00 | 0.89 | 0.94 | 257^{*} | 0.50 | 0.06 | 0.11 |
| 221 | 1.00 | 1.00 | 1.00 | 257-2* | 1.00 | 0.97 | 0.98 |
| 222 | 1.00 | 1.00 | 1.00 | 257-4* | 0.94 | 0.88 | 0.91 |
| 223 | 0.98 | 0.98 | 0.98 | 257-6* | 0.84 | 0.79 | 0.81 |
| 224 | 1.00 | 1.00 | 1.00 | 257-8* | 0.89 | 0.76 | 0.82 |
| 225 | 1.00 | 1.00 | 1.00 | 258^{*} | 0.80 | 0.60 | 0.69 |
| 228 | 1.00 | 1.00 | 1.00 | 258-2* | 0.97 | 0.94 | 0.95 |
| 230 | 0.94 | 1.00 | 0.97 | 258-4* | 0.96 | 0.88 | 0.92 |
| 231 | 1.00 | 1.00 | 1.00 | 258-6* | 0.95 | 0.82 | 0.88 |
| 232 | 1.00 | 1.00 | 1.00 | $258 - 8^*$ | 0.94 | 0.78 | 0.85 |
| 233 | 1.00 | 1.00 | 1.00 | 259^{*} | 0.89 | 0.70 | 0.78 |
| 236 | 1.00 | 1.00 | 1.00 | 259-2* | 0.98 | 0.95 | 0.96 |
| 237 | 1.00 | 1.00 | 1.00 | 259-4* | 0.98 | 0.95 | 0.96 |
| 238 | 0.99 | 0.99 | 0.99 | 259-6* | 0.98 | 0.95 | 0.96 |
| 239 | 0.97 | 1.00 | 0.98 | 259-8* | 0.98 | $0.00 \\ 0.95$ | 0.96 |
| $\frac{200}{240}$ | 0.97 | 1.00 | 0.98 | 260×10^{-2} | 0.94 | $0.50 \\ 0.55$ | 0.60 |
| $240 \\ 2/1$ | 1.00 | 1.00 | 1.00 | 260-2* | 0.94 | 0.00 | 0.03 |
| 241 | 1.00 0.07 | 1.00 1.00 | 0.08 | 260-2 260-4* | 0.00 | 0.00 | 0.04 |
| $240 \\ 247$ | 0.01 | 1.00 0.07 | 0.95 | 260-4 260-6* | 0.55 | 0.55 0.70 | 0.33 0.87 |
| $\frac{241}{248*}$ | 1.00 | $\frac{0.91}{0.81}$ | 0.00 | $-\frac{200-0}{260-8*}$ | 0.50 | $0.13 \\ 0.72$ | 0.01 |
| 240 | 1.00 | 0.01 | 0.30 | 261* | 0.00 0.67 | 0.12 | 0.15 |
| 240-2 | 1.00 | 0.95 | 0.97 | 201 | 0.07 | 0.40 | 0.50 |
| 240-4 | 1.00 1.00 | 0.92 0.88 | 0.90 | 201-2 | 0.88 | 0.91 0.01 | 0.89 |
| 240-0 | 1.00 | 0.85 | 0.94 | 201-4 | 0.88 | 0.91 | 0.89 |
| 240-0 | 0.90 | 0.00 | 0.91 | 201-0 | 0.88 | 0.91 | 0.89 |
| 249 | 0.00 | 0.00 | 0.74 | 201-0 | 0.00 | 0.91 | 0.89 |
| $249-2^{+}$ | 0.90 | 0.95 | 0.90 | 202 | 0.00 | 0.00 | 0.00 |
| $249-4^{\circ}$ | 0.98 | 0.91 | 0.94 | $202-2^{+}$ | 1.00 | 0.79 | 0.88 |
| 249-0 240 0* | 0.98 | 0.87 | 0.92 | 202-4 | 1.00 | 0.01 | 0.70 |
| 249-8" 250* | 0.95 | 0.82 | 0.88 | 202-0* | 1.00 | 0.42 | 0.59 |
| 200" 250.0* | 0.90 | 0.58 | 0.71 | 202-8* | 1.00 | 0.21 | 0.35 |
| 200-27 | 1.00 | 1.00 | 1.00 | 200 ⁺⁺ | 0.80 | 0.14 | 0.24 |
| $250-4^{+}$ | 1.00 | 1.00 | 1.00 | 200* | 0.30 | 0.09 | 0.14 |
| 250-6 [↑] | 1.00 | 1.00 | 1.00 | 301 | 0.94 | 0.82 | 0.88 |
| 250-8* | 1.00 | 0.88 | 0.94 | 302 | 0.89 | 0.65 0.71 | 0.75 |
| | | | | 303 304 | $0.00 \\ 0.05$ | 0.11 0.07 | 0.08 |
| | | | | 07004 | 0.30 | 0.31 | 0.30 |

Table 7: Matching results on benchmark2008

| | | easure | 00 | 97 | 66 | 83 | 82 | 88 | | easure | 00 | 97 | 00 | 79 | 62 | 85 | | easure | 6 | 7 | C | ∞ | 5 | 2 |
|---------------|---------|-------------------|---------|---------|-----------|---------|-----------|------|-------|------------|---------|---------|-----------|---------|-----------|------|--------------|------------|---------|-----------|-----------|----------|-----------|------|
| | V | F1-m | 1. | 0. | 0. | .0 | 0 | 0. | M | F1-m | 1. | 0. | 1. | 0. | 0. | 0. | lap | F1-m | 0.6! | $0.2^{'}$ | 0.9(| 0.38 | 0.5' | 0.4(|
| | Lily | Recall | 1.00 | 0.95 | 1.00 | 0.76 | 0.79 | 0.84 | RiMC | Recall | 1.00 | 0.95 | 1.00 | 0.70 | 0.80 | 0.80 | LogM | Recall | 0.69 | 0.21 | 0.97 | 0.33 | 0.38 | 0.43 |
| | | Precision | 1.00 | 1.00 | 0.99 | 0.92 | 0.86 | 0.94 | | Precision | 1.00 | 0.99 | 1.00 | 0.92 | 0.79 | 0.94 | | Precision | 0.68 | 0.44 | 0.84 | 0.50 | 0.93 | 0.57 |
| | lood | F1-measure | 1.00 | 0.87 | 0.99 | 0.69 | 0.65 | 0.76 | 0 | F1-measure | 0.96 | 0.47 | 0.91 | 0.42 | 0.21 | 0.51 | | F1-measure | 0.73 | 0.53 | 0.86 | 0.50 | 0.51 | 0.57 |
| | chor-F. | Recall | 1.00 | 0.80 | 1.00 | 0.56 | 0.61 | 0.69 | MapPS | Recall | 1.00 | 0.50 | 0.96 | 0.41 | 0.21 | 0.53 | AML | Recall | 0.72 | 0.43 | 0.86 | 0.41 | 0.39 | 0.49 |
| 08 | An | Precision | 1.00 | 0.97 | 0.99 | 0.89 | 0.71 | 0.92 | I | Precision | 0.92 | 0.46 | 0.86 | 0.43 | 0.22 | 0.51 | | Precision | 0.75 | 1.00 | 0.88 | 0.81 | 0.85 | 0.85 |
| enchmark 20 | l | F1-measure | 1.00 | 0.84 | 0.99 | 0.65 | 0.78 | 0.73 | ſe | F1-measure | 0.87 | 0.76 | 0.81 | 0.50 | 0.41 | 0.58 | ap | F1-measure | 0.51 | 0.22 | 0.76 | 0.34 | 0.30 | 0.37 |
| ms on b | DSSin | \mathbf{Recall} | 1.00 | 0.75 | 1.00 | 0.51 | 0.69 | 0.65 | GeRoN | Recall | 0.79 | 0.69 | 0.73 | 0.49 | 0.37 | 0.57 | TaxoM | Recall | 0.34 | 0.13 | 0.62 | 0.23 | 0.19 | 0.28 |
| of 15 syste | | Precision | 1.00 | 0.96 | 0.99 | 0.90 | 0.89 | 0.93 | | Precision | 0.96 | 0.85 | 0.92 | 0.51 | 0.46 | 0.64 | | Precision | 1.00 | 0.67 | 0.99 | 0.67 | 0.70 | 0.73 |
| ing results c | V | F1-measure | 1.00 | 0.97 | 0.99 | 0.80 | 0.76 | 0.85 | ~ | F1-measure | 0.99 | 0.76 | 0.97 | 0.59 | 0.81 | 0.68 | dtf | F1-measure | 1.00 | 0.71 | 0.99 | 0.59 | 0.85 | 0.66 |
| ll matchi | ASMO | Recall | 1.00 | 0.96 | 1.00 | 0.73 | 0.75 | 0.82 | CIDEF | Recall | 0.99 | 0.65 | 0.98 | 0.48 | 0.73 | 0.61 | AMBO | Recall | 1.00 | 0.55 | 0.99 | 0.44 | 0.79 | 0.58 |
| 8: Overa | | Precision | 1.00 | 0.98 | 0.99 | 0.90 | 0.78 | 0.93 | | Precision | 0.99 | 0.92 | 0.97 | 0.76 | 0.90 | 0.83 | \mathbf{S} | Precision | 1.00 | 0.98 | 0.99 | 0.87 | 0.92 | 0.91 |
| Table | сų. | F1-measure | 0.99 | 0.76 | 0.97 | 0.59 | 0.31 | 0.66 | A | F1-measure | 1.00 | 0.91 | 0.97 | 0.66 | 0.74 | 0.75 | 0 | F1-measure | 1.00 | 0.68 | 0.99 | 0.59 | 0.86 | 0.65 |
| | SPIDE | Recall | 0.99 | 0.65 | 0.98 | 0.48 | 0.80 | 0.61 | AROM | Recall | 1.00 | 0.86 | 0.98 | 0.54 | 0.68 | 0.68 | SAMB(| Recall | 1.00 | 0.52 | 0.99 | 0.44 | 0.79 | 0.58 |
| | | Precision | 0.99 | 0.92 | 0.97 | 0.76 | 0.19 | 0.80 | | Precision | 1.00 | 0.98 | 0.97 | 0.83 | 0.81 | 0.88 | | Precision | 1.00 | 0.98 | 0.99 | 0.87 | 0.95 | 0.91 |
| | | | 101-104 | 201-210 | 221 - 247 | 248-266 | 301 - 304 | Avg. | | I | 101-104 | 201-210 | 221 - 247 | 248-266 | 301 - 304 | Avg. | | | 101-104 | 201-210 | 221 - 247 | 248-266 | 301 - 304 | Avg. |

| | | sasure | 0 | 4 | 6 | 7 | | × | | sasure | 0 | 4 | 6 | _ | | ~ | | sasure | | ~1 | \sim | | \sim | _ |
|--------------|--------|-------------------|---------|---------|-----------|---------|-----------|------|--------------|-------------------|---------|---------|-----------|-----------|-----------|------|--------|----------------|---------|---------|-----------|---------|-----------|------|
| | | F1-mé | 1.0 | 0.9 | 0.9 | 0.8 | 0.8(| 0.8 | M | F1-mé | 1.0 | 0.9 | 0.9 | 0.7_{4} | 0.8(| 0.85 | ap | $F1-m\epsilon$ | 0.92 | 0.35 | 0.75 | 0.40 | 0.55 | 0.47 |
| | Lily | Recall | 1.00 | 0.95 | 1.00 | 0.76 | 0.79 | 0.84 | RiMO | Recall | 1.00 | 0.95 | 1.00 | 0.69 | 0.80 | 0.79 | LogM | Recall | 0.92 | 0.26 | 0.76 | 0.36 | 0.39 | 0.43 |
| | | Precision | 1.00 | 0.99 | 0.99 | 0.93 | 0.83 | 0.95 | | Precision | 1.00 | 0.99 | 0.99 | 0.84 | 0.80 | 0.89 | | Precision | 0.91 | 0.50 | 0.73 | 0.52 | 0.93 | 0.58 |
| | lood | F1-measure | 1.00 | 0.93 | 0.99 | 0.77 | 0.83 | 0.84 | 0 | F1-measure | 1.00 | 0.90 | 0.97 | 0.39 | 0.25 | 0.58 | | F1-measure | 0.98 | 0.46 | 0.98 | 0.49 | 0.51 | 0.58 |
| | chor-F | \mathbf{Recall} | 1.00 | 0.89 | 1.00 | 0.70 | 0.78 | 0.79 | <u>MapPS</u> | \mathbf{Recall} | 1.00 | 0.90 | 0.98 | 0.39 | 0.24 | 0.58 | AML | Recall | 0.96 | 0.36 | 0.96 | 0.40 | 0.39 | 0.50 |
| 600 | An | Precision | 1.00 | 0.98 | 0.99 | 0.96 | 0.90 | 0.97 | | Precision | 1.00 | 0.90 | 0.97 | 0.39 | 0.27 | 0.58 | | Precision | 1.00 | 0.89 | 0.99 | 0.80 | 0.85 | 0.85 |
| oenchmark 2 | n | F1-measure | 1.00 | 0.78 | 1.00 | 0.62 | 0.74 | 0.72 | Ie | F1-measure | 1.00 | 0.88 | 0.99 | 0.66 | 0.63 | 0.76 | ap | F1-measure | 0.51 | 0.26 | 0.71 | 0.37 | 0.43 | 0.41 |
| ems on 1 | DSSin | Recall | 1.00 | 0.70 | 1.00 | 0.52 | 0.65 | 0.64 | GeRoN | Recall | 1.00 | 0.86 | 1.00 | 0.58 | 0.58 | 0.71 | TaxoM | Recall | 0.34 | 0.17 | 0.63 | 0.30 | 0.31 | 0.33 |
| of 15 syst | | Precision | 1.00 | 0.95 | 0.99 | 0.91 | 0.92 | 0.93 | | Precision | 1.00 | 0.92 | 0.99 | 0.81 | 0.71 | 0.86 | | Precision | 1.00 | 0.65 | 0.97 | 0.64 | 0.77 | 0.71 |
| ning results | V | F1-measure | 1.00 | 0.97 | 0.99 | 0.79 | 0.79 | 0.86 | ter | F1-measure | 0.98 | 0.76 | 0.98 | 0.54 | 0.83 | 0.67 | M | F1-measure | 0.98 | 0.43 | 0.96 | 0.44 | 0.60 | 0.54 |
| all match | ASMO | Recall | 1.00 | 0.96 | 1.00 | 0.73 | 0.80 | 0.82 | AgrMak | \mathbf{Recall} | 0.98 | 0.70 | 0.99 | 0.45 | 0.77 | 0.60 | SOBOI | Recall | 0.97 | 0.35 | 0.96 | 0.33 | 0.52 | 0.46 |
| e 9: Over | | Precision | 1.00 | 0.98 | 0.99 | 0.90 | 0.80 | 0.93 | | Precision | 0.98 | 0.99 | 0.98 | 0.87 | 0.92 | 0.92 | | Precision | 0.98 | 0.72 | 0.96 | 0.78 | 0.86 | 0.81 |
| Table | | F1-measure | 0.98 | 0.51 | 0.76 | 0.35 | 0.59 | 0.47 | A | F1-measure | 1.00 | 0.90 | 0.97 | 0.61 | 0.79 | 0.73 | d | F1-measure | 0.99 | 0.69 | 0.96 | 0.54 | 0.56 | 0.65 |
| | enda | Recall | 1.00 | 0.52 | 1.00 | 0.45 | 0.80 | 0.58 | AROM | Recall | 1.00 | 0.85 | 0.98 | 0.52 | 0.76 | 0.67 | kosima | Recall | 0.99 | 0.61 | 0.97 | 0.45 | 0.48 | 0.58 |
| | | Precision | 0.96 | 0.50 | 0.67 | 0.31 | 0.47 | 0.42 | | Precision | 1.00 | 0.98 | 0.97 | 0.84 | 0.83 | 0.89 | | Precision | 0.99 | 0.88 | 0.96 | 0.86 | 0.69 | 0.88 |
| | | | 101-104 | 201-210 | 221 - 247 | 248-266 | 301 - 304 | Avg. | | I | 101-104 | 201-210 | 221 - 247 | 248-266 | 301 - 304 | Avg. | | | 101-104 | 201-210 | 221 - 247 | 248-266 | 301 - 304 | Avg. |

Table 7 presents detailed matching results on benchmark2008 by Lily. Matching task numbers with star are weak informative matching tasks. Lily performs very well on the informative ontology matching tasks. For group 101-104 and 221-247, matching results are perfect and average precision, recall and F1-measure are all 0.99. For the real-world group 301-304, Lily also obtains good results with average 0.86 precision, 0.79 recall and 0.82 F1-measure. For most weak informative ontologies matching tasks in group 201-210 and 248-266, Lily also gets good results. We notice that matching results on 254, 257, 261, 262, 262-6, 262-8, 265 and 266 have less than 0.6 F1-measure. The reason is that not only the ontology structures are suppressed, but also almost all meaningful literal information is removed.

Table 8 shows results of 15 matching systems on benchmark2008 [22, 34], which is also divided into five groups: 101-104, 201-210, 221-247, 248-266 and 301-304. Here the average results on each group are presented in a row, and the last row shows the overall average results for all matching tasks. (1) Group 101-104 contains some simple matching tasks, except GeRoMe and TaxoMap, other 12 systems obtain good results, and 8 systems including Lily have 1.00 F1-measure. (2) For group 201-210, since some ontologies are weak informative, some systems do not perform well. F1-measure of 6 systems is higher than 0.80. F1-measure of 4 systems is higher than 0.90. The best three systems: ASMOV, RiMOM and Lily get 0.97 F1-measure. (3) For group 221-247, 11 systems perform very well and have F1-measure of higher than 0.90. RiMOM has the perfect result with 1.00 F1-measure. Lily has similar results with 0.99 F1-measure. (4) For group 248-266, most systems cannot deal with these tasks efficiently. Only 3 systems have higher than 0.70 F1-measure, and they are ASMOV, Lily and RiMOM. Only Lily and ASMOV have 0.80 and 0.82 F1-measure, respectively. Lily has 0.92 precision and 0.76 recall, which are the best results in all systems. Lily performs best on this difficult group and obtains better results than most systems. (5) For group 301-304, not all systems can perform well on the real-world matching tasks, and only 4 systems have higher than 0.8 F1-measure. Best F1-measure is 0.86 by SAMBO. Lily has 0.82 F1measure. For the overall average results, only 3 systems, ASMOV, RiMOM and Lily, have higher than 0.80 F1-measure. That dues to good performance on weak informative ontology matching tasks. Lily has average 0.88 F1-measure, 0.94 precision and 0.84 recall, all of them are the best in all systems.

Table 9 shows results of 15 systems on benchmark2009 [35, 36]. Lily is also one of the best systems. Especially for group 248-266, Lily still obtains best results. For the overall average results, 4 systems have average higher than 0.8 F1-measure: ASMOV, Lily, Anchor-Flood and RiMOM. Lily has the best F1-measure of 0.88 and best recall of 0.84. Lily also has 0.95 precision, only lower than 0.97 precision by Anchor-Flood.

11. Related Work

Many ontology matching techniques have been proposed. Some researchers have given very comprehensive surveys for this open problem [4, 5, 3, 7, 6, 8, 9]. In recent years, this problem still attracts the attentions of researchers, and new ontology matching methods are proposed. Zhao et al. proposed a method based on formal concept analysis to identify and

validate mappings across ontologies [37]. Faria et al. discussed the methodology for automatically selecting background knowledge sources for any given ontologies to match [38]. The simulated annealing, which is a evolutionary algorithm, was also used to find the mappings between two given ontologies while no ground truth is available [39]. However, these works ignore the weak informative ontologies in a lot of practical applications. This paper mainly focuses on the weak informative ontology matching problem. Structure-based method is a feasible way to deal with this special matching problem. For the reason that the ontology graph topology cannot represent the semantics reasonably, methods using classical graph matching algorithm cannot work and has unsatisfied matching performance. Therefore, the method based on similarity propagation idea is the practical way to solve the problem.

Blondel et al. proposed a iteration equation for measuring the similarity between directed graphs [19]. It is based on the Hub-Authority idea, which is a kind of similarity propagation. Leicht et al. proposed a more general measurement for the vertex similarity in network [31], they also pointed out that Blondel's method is a special case of their method. However, the two graph matching methods can only calculate the vertex similarity in graph, but cannot deal with the edge similarity. To overcome this shortcoming, Hu et al. used the bipartite graph to represent the ontology graph, and then they modified Blondel's method to handle ontology graph matching problem [30]. This method is implemented in Falcon-AO [15]. Inspired by these work, Tous and Delgado first represented ontology graphs as the vector space, then used Blondel's graph matching algorithm to matching weak informative ontologies [40] and obtained close performance to Falcon-AO.

Similarity flooding [17] is the most popular algorithm inspired by the similarity propagation, but it cannot be directly used for ontology matching (see the reasons we discussed in previous sections). However, some weak informative ontology matching methods are based on similarity flooding.

Noy and Musen proposed AnchorPrompt algorithm [29] to find alignment between concepts by analyzing the ontology graph structure. The principle of AnchorPrompt is that if two pairs of elements from different ontologies are similar and there are paths connecting the elements, then elements in those paths are often similar as well. Therefore, it is also a variation of the similarity flooding idea. However, this approach does not work well when ontologies are constructed in different ways. For example, one ontology has a deep concept hierarchy with many inter-linked concepts and another ontology is a shallow one where the hierarchy has only a few levels.

In ontology mapping system RiMOM [16], Li and Tang et al. used three propagation strategies for structure matching: (1) propagation between concepts; (2) propagation between relations; (3) propagation between concepts and relations. This method employs ontological structure information to design propagation for matching ontologies. It is a very successful variation of similarity flooding for ontology matching.

In ASMOV [41], Jean-Mary et al. used two structural similarities: a relational similarity and an internal similarity, to discover alignments. The relational similarity is computed by combining the similarities between parents and children. The internal similarity is calculated by considering domain and range of properties, property restrictions associated to a concept and other structure information. These structural similarities are calculated iteratively, and also use similarities propagated by neighbors. Therefore, it can be seen as a special case of similarity flooding.

Seddiqui and Aono proposed Anchor-Flood algorithm [42]. This algorithm first uses few initial correspondences and gradually explored concepts by collecting concept neighbors and other graph data structure to output a set of correspondences between concepts and properties in semantically connected subsets. The process runs iteratively until the algorithm satisfies the condition: *either all the collected concepts are explored, or no new aligned pair is found.* This method is also inspired by the idea of similarity flooding.

In AgreementMaker [43], Cruz et al.used a structure-based matcher called DSI (Descendant's Similarity Inheritance). DSI is based on the heuristic rule: if two nodes are matched with high similarity, then the similarity between the descendants of those nodes should increase. It is also a kind of variation of similarity flooding. DSI matcher is also used by another system CSA [44].

In SAMBO [45], Lambrix and Tan proposed a structural matcher, which was an iterative algorithm based on the *is-a* and *part-of* hierarchies of ontologies. The algorithm is based on the intuition: if two concepts are in similar positions with respect to *is-a* or *part-of* hierarchies and are relative to already aligned concepts, then they are likely to be similar as well.

In GeRoMe [46, 47], Quix et al. also found that direct Similarity Flooding had no positive effect on the matching quality. Meanwhile, this matching system designed structural similarity called children and parent matchers, which propagate the similarity of elements up and down in the class hierarchy.

In PRIOR+ [48], Mao et al. did not directly use similarity flooding, but they proposed an important similarity called profile similarity. This method first generates a profile for each element using label, comments and other related literal information of the element, then the profile of the element's ancestors, descendants and siblings will be passed to that of the element with different weights. Finally, the cosine similarity between the profiles of two elements is calculated in a vector space model. We believe this is also a kind of variation of similarity flooding, because the profile propagation is a kind of similarity propagation.

In LogMap [25, 26], Jiménez-Ruiz et al. realized a similar propagation idea. First, LogMap computes an initial set of equivalence anchor mappings by intersecting the lexical indexes of each input ontology. Then these mappings will later serve as starting point for the further discovery of additional mappings. This mapping discovery strategy is based on a principle: if classes C_1 and C_2 are correctly mapped, then the classes semantically related to C_1 in O_1 are likely to be mapped to those semantically related to C_2 in O_2 .

Similarity flooding idea is also used in DSSim [49] and YAM++ [50]. YAM++ proposed the confidence propagation based on similarity flooding [51]. The intuition behind the confidence propagation is that the similarity of the two matched concepts will contribute in some degree of confidence to the similarity of their relatives along the same path of semantic relations.

Since some weak informative ontologies in OAEI benchmark datasets can be seen as different versions of the same ontology, CODI [52] first matches different ontology versions, then refines the matching results by markov logic framework [53]. However, this method also can be used in matching tasks not generated by different versions.

Similar problem also arises in schema matching in database research field. For example, the column names in the schemas and the data in the columns are opaque or very difficult to interpret. Kang and Naughton presented two-step technique that works even in the presence of opaque column names and data values [54]. They constructed a dependency graph using mutual information, then found matching node pairs in dependency graphs by running a graph matching algorithm. However, it is an instance-based matcher and cannot solve the concept matching or property matching in ontologies.

Compared to current works, our method has some features: (1) It is the first general similarity propagation framework for ontology matching instead of some heuristic propagation rules, and it can discover alignments of concepts, properties and instances. (2) It uses semantic subgraphs to capture precise meanings of ontology elements, then constructs semantic description documents, finally, designs a matcher to provide few credible correspondences as seeds for similarity propagation model. (3) It utilizes the strong constraint condition and semantic subgraphs to restrict the range of similarity propagation, that can avoid the meaningless propagations. Meanwhile, our method also has limitations: (1) First, extracting semantic subgraphs for all elements is a time-consuming process, in which solving the linear equations in the circuit model is the time bottleneck. Fortunately, this limitation can be solved by parallel computing. (2) Second, there are more parameters (such as semantic graph size) and settings (such as propagation strategy) in our method, it needs more experiences to tune the parameters and select reasonable settings. In our ongoing work, we are designing the automatic tuning method to help users to select optimized parameters and settings.

12. Conclusion

This paper proposes a method for matching weak informative ontologies. The success of our method is due to two techniques: semantic subgraphs and a new similarity propagation model. Semantic subgraphs can precisely describe ontology elements with limited triples. The similarity propagation model is based on the strong constrained condition, and it is reasonable for handling ontology matching. Some useful propagation strategies are also adopted to improve matching results. Our method is used for discovering correspondences between concepts or properties. However, in recent years, knowledge base such as knowledge graph [55] usually contains large scale instances, which would also be weak informative. In the very near future, we will extend our method for matching large scale knowledge graphs. In addition, except using WordNet to define weak informative ontology, we do not use any external knowledge or data in ontology matching. In the future, we will enhance the matcher based on semantic subgraphs using word2vec [56] and knowledge graph embedding [57], which are trained on external corpus and knowledge graphs respectively. That would provide more credible seeds for similarity propagation.

References

- [1] A. Doan, J. Madhavan, P. Domingos, A. Halevy, Learning to map between ontologies on the semantic web, in: Proceedings of the 11th international conference on World Wide Web (WWW2002)., 2002.
- [2] D. Aumueller, H.-H. Do, S. Massmann, E. Rahm, Schema and ontology matching with coma++, in: Proceedings of the 2005 ACM SIGMOD international conference on management of data., 2005.
- [3] E. Rahm, P. A. Bernstein, A survey of approaches to automatic schema matching, The VLDB Journal 10 (2001) 334–350.
- [4] Y. Kalfoglou, M. Schorlemmer, Ontology mapping: the state of the art, The Knowledge Engineering Review 18(1) (2003) 1–31.
- [5] P. Shvaiko, J. Euzenat, A survey of schema-based matching approaches, Journal on Data Semantics 4 (2005) 146–171.
- [6] P. Shvaiko, J. Euzenat, Ontology matching: state of the art and future challenges, IEEE Transactions on Knowledge and Data Engineering 25(1) (2011) 158–176.
- [7] J. Euzenat, P. Shvaiko, Ontology matching (2nd edition), Heidelberg: Springer-Verlag, 2013.
- [8] L. Otero-Cerdeira, F. J. Rodríguez-Martínez, A. Gómez-Rodríguez, Ontology matching: a literature review, Expert Systems with Applications 42(2) (2015) 949–971.
- P. Ochieng, S. Kyanda, Large-scale ontology matching: state-of-the-art analysis, ACM Computing Surveys 51(4) (2018) 1–35.
- [10] D. Pfisterer, K. Römer, D. Bimschas, et al., Spitfire: toward a semantic web of things, IEEE Communications Magazine 49(11) (2011) 40–48.
- [11] Y. Tang, T. Liu, G. Liu, et al., Enhancement of power equipment management using knowledge graph, in: Proceedings of the IEEE Innovative Smart Grid Technologies-Asia . Chengdu, China, 2019.
- [12] Y. Huang, X. Zhou, Knowledge model for electric power big data based on ontology and semantic web, CSEE Journal of Power and Energy Systems 1(1) (2015) 19–27.
- [13] S. R. Bader, I. Grangel-Gonzalez, P. Nanjappa, et al., A knowledge graph for industry 4.0, in: Proceedings of the 17th European Semantic Web Conference (ESWC2020). Heraklion, Crete, Greece, 2020.
- [14] D. Conte, P. Foggia, C. Sansone, et al., Thirty years of graph matching in pattern recognition, International Journal of Pattern Recognition and Artificial Intelligence 18(3) (2004) 265–298.
- [15] W. Hu, Y. Qu, Falcon-ao: A practical ontology matching system, Journal of Web Semantics (2008) 237–239.
- [16] J. Li, J. Tang, Y. Li, et al., Rimom: a dynamic multistrategy ontology alignment framework, IEEE Transactions on Knowledge and Data Engineering 21(8) (2009) 1218–1232.
- [17] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity flooding: a versatile graph matching algorithm and its application to schema matching, in: Proceeding of the 18th International Conference on Data Engineering (ICDE2002). San Jose, CA, USA, 2002.
- [18] G. Jeh, J. Widom, Simrank: a measure of structural-context similarity, in: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2002). Edmonton, Alberta, Canada, 2002.
- [19] V. D. Blondel, A. Gajardo, M. Heymans, et al., A measure of similarity between graph vertices: applications to synonym extraction and web searching, SIAM Review 46(4) (2004) 647–666.
- [20] A. Doan, A. Y. Halevy, Semantic integration research in the database community: a brief survey, AI magazine 26(1) (2005) 83–94.
- [21] P. Wang, B. Xu, An effective similarity propagation model for matching ontologies without sufficient or regular linguistic information, in: Proceedings of the 4th Asian Semantic Web Conference (ASWC2009). Shanghai, China, 2009.
- [22] C. Caraciolo, J. Euzenat, L. Hollink, et al., Results of the ontology alignment evaluation initiative 2008, in: Proceedings of the 3rd ISWC Workshop on Ontology Matching (OM2008). Karlsruhe, Germany, 2008.
- [23] D. Faria, C. Pesquita, E. Santos, et al., The agreementmakerlight ontology matching system, in: Proceedings of the 12th On the Move to Meaningful Internet Systems (OTM2013), Graz, Austria, 2013.
- [24] D. Faria, C. Pesquita, B. S. Balasubramani, et al., Results of aml participation in oaei 2018, in:

Proceedings of the 12th International Workshop on Ontology Matching (OM2018), Monterey, CA, USA, 2018.

- [25] E. Jiménez-Ruiz, B. C. Grau, Logmap: Logic-based and scalable ontology matching, in: Proceedings of the 10th International Semantic Web Conference (ISWC2011), Bonn, Germany, 2011.
- [26] E. Jiménez-Ruiz, B. C. Grau, V. Cross, Logmap family participation in the oaei 2018, in: Proceedings of the 12th International Workshop on Ontology Matching (OM2018), Monterey, CA, USA, 2018.
- [27] P. Ziegler, C. Kiefer, C. Sturm, K. R. Dittrich, A. Bernstein, Generic similarity detection in ontologies with the soqa-simpack toolkit, in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD2006). Chicago, Illinois, USA, 2006.
- [28] M. Ehrig, S. Staab, Qom quick ontology mapping, in: Proceedings of the 3rd International Semantic Web Conference (ISWC2004). Hiroshima, Japan, 2004.
- [29] N. F. Noy, M. A. Musen, The prompt suite: interactive tools for ontology merging and mapping, International Journal of Human-Computer Studies 59(6) (2003) 983–1024.
- [30] W. Hu, N. Jian, Y. Qu, et al., Gmo: a graph matching for ontologies, in: Proceedings of K-CAP Workshop on Integrating Ontologies. Banff, Alberta, Canada, 2005.
- [31] E. A. Leicht, P. Holme, M. E. J. Newman, Vertex similarity in networks, Physical Review E 73.
- [32] C. Faloutsos, K. S. McCurley, A. Tomkins, Fast discovery of connection subgraphs, in: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2004). Seattle, Washington, USA., 2004.
- [33] Y. Qu, W. Hu, G. Cheng, Constructing virtual documents for ontology matching, in: Proceedings of the 15th International Conference on World Wide Web (WWW2006). Edinburgh, Scotland, UK, 2006.
- [34] P. Wang, B. Xu, Lily: ontology alignment results for oaei 2008, in: Proceedings of the 3rd International Workshop on Ontology Matching (OM2008). Karlsruhe, Germany, 2008.
- [35] J. Euzenat, A. Ferrara, L. Hollink, et al., Results of the ontology alignment evaluation initiative 2009, in: Proceedings of the 4th ISWC Workshop on Ontology Matching (OM2009). Trento, Italy, 2009.
- [36] P. Wang, B. Xu., Lily: ontology alignment results for oaei 2009, in: Proceedings of the 4th International Workshop on Ontology Matching (OM2009). Chantilly, VA, USA, 2009.
- [37] M. Zhao, S. Zhang, W. Li, G. Chen, Matching biomedical ontologies based on formal concept analysis, Journal of Biomedical Semantics 9(11) (2018) 1–27.
- [38] D. Faria, C. Pesquita, E. Santos, I. F. Cruz, F. M. Couto, Automatic background knowledge selection for matching biomedical ontologies, PLoS ONE 9(11) (2014) e111226.
- [39] M. Mohammadi, W. Hofman, Y.-H. Tan, Sanom-hobbit: simulated annealing-based ontology matching on hobbit platform, The Knowledge Engineering Review 35(13) (2020) 1–13.
- [40] R. Tous, J. Delgado, A vector space model for semantic similarity calculation and owl ontology alignment, in: Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA2006). Krakow, Poland, 2006.
- [41] Y. R. Jean-Mary, E. P. Shironoshita, M. R. Kabuka, Ontology matching with semantic verification, Journal of Web Semantics 7(3) (2009) 235–251.
- [42] M. H. Seddiqui, M. Aono, An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size, Journal of Web Semantics 7(4) (2009) 344–356.
- [43] I. F. Cruz, F. P. Antonelli, C. Stroe, Agreementmaker: efficient matching for large real-world schemas and ontologies, PVLDB 2(2) (2009) 1586–1589.
- [44] Q.-V. Tran, R. Ichise, B.-Q. Ho, Cluster-based similarity aggregation for ontology matching, in: Proceedings of the 6th International Workshop on Ontology Matching (OM2011). Bonn, Germany, 2011.
- [45] P. Lambrix, H. Tan, Sambo a system for aligning and merging biomedical ontologies, Journal of Web Semantics 4 (2006) 196–206.
- [46] D. Kensche, C. Quix, L. X, et al., Geromesuite: A system for holistic generic model management, in: Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB2007). Vienna, Austria, 2007.
- [47] C. Quix, S. Geisler, D. Kensche, et al., Results of geromesuite for oaei 2009, in: Proceedings of the 4th International Workshop on Ontology Matching (OM2009). Chantilly, VA, USA, 2009.

- [48] M. Mao, Y. Peng, M. Spring, An adaptive ontology mapping approach with neural network based constraint satisfaction, Journal of Web Semantics 8(1) (2010) 14–25.
- [49] M. Nagy, M. Vargas-Vera, Towards an automatic semantic data integration: multi-agent framework approach, Semantic Web (2010) 107–134.
- [50] D. Ngo, Z. Bellahsene, R. Coletta, Yam++ results for oaei 2011, in: Proceedings of the 6th International Workshop on Ontology Matching (OM2011). Bonn, Germany, 2011.
- [51] D. Ngo, Z. Bellahsene, Overview of yam++--(not) yet another matcher for ontology alignment task, Journal of Web Semantics 41 (2016) 30–49.
- [52] M. Niepert, C. Meilicke, H. Stuckenschmidt, A probabilistic-logical framework for ontology matching, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI2010). Georgia, USA, 2010.
- [53] J. Huber, T. Sztyler, J. Noessner, et al., Codi: combinatorial optimization for data integration results for oaei 2011, in: Proceedings of the 6th International Workshop on Ontology Matching (OM2011). Bonn, Germany, 2011.
- [54] J. Kang, J. F. Naughton, On schema matching with opaque column names and data values, in: Proceedings of the 2003 ACM SIGMOD international conference on Management of data (SIGMOD2003). California, USA, 2003.
- [55] X. Dong, E. Gabrilovich, G. Heitz, et al., Knowledge vault: a web-scale approach to probabilistic knowledge fusion, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2014). New York, USA, 2014.
- [56] T. Mikolov, I. Sutskever, K. Chen, et al., Distributed representations of words and phrases and their compositionality, in: Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS2013). Nevada, USA, 2013.
- [57] A. Bordes, N. Usunier, A. Garcia-Duran, et al., Translating embeddings for modeling multi-relational data, in: Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS2013). Nevada, USA, 2013.