JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 31, 415-432 (2015)

Ontology Kernel – A Convolution Kernel for Ontology Alignment*

JEONG-WOO SON[§], HEE-GUEN YOON[‡] AND SEONG-BAE PARK[‡,+]
[§]*Broadcasting and Telecommunications Media Research Laboratory*
Electronics and Telecommunications Research Institute
Daejeon, 305-700 Korea
E-mail: jwson@etri.re.kr
[‡]*School of Computer Science and Engineering*
Kyungpook National University
Daegu, 702-701 Korea
E-mail: {hkyoon; sbpark}@sejong.knu.ac.kr

Every ontology entity such as a concept or a property has its own structural information represented as a graph due to the relations with other entities. Therefore, it is important to consider not only its lexical similarity but also structural similarity in ontology alignment. This paper proposes ontology kernel that computes both types of similarities simultaneously. The idea of this kernel is to measure the structural similarity of ontology entities by mapping their entity graphs into the space spanned by entity random walks. The graph of an entity in the kernel expresses all relations with other entities. Thus, the ontology kernel can compare the similarity between entities no matter how complex the entities are and no matter how many kinds of relations they possess. A series of experiments with the standard data sets prove the generality and the superiority of the ontology kernel in ontology alignment.

Keywords: ontology alignment, ontology kernel, convolution kernel, graph similarity, hyper graph

1. INTRODUCTION

Ontology is an explicit, machine-readable specification of a shared conceptualization of a domain. It can be an information source for various applications. Especially in Semantic Web, an ontology is essential not only as an information source but also for supporting inter-operability among applications. One of the issues related with ontologies is that most ontologies are written separately and independently by human experts to serve particular tasks or application domains. Thus, there are many ontologies even in a single domain, and it causes semantic heterogeneity. This heterogeneity of ontologies for a domain becomes an obstacle for inter-operability of applications. Therefore, a method to merge the related ontologies is required to restore the inter-operability.

Ontology alignment aims to merge two ontologies which contain similar semantic information by identifying semantic similarities between entities in the ontologies. Thus,

Received August 19, 2013; revised December 18, 2013 & February 13, 2014; accepted March 17, 2014.
Communicated by Chang-Shing Lee.
* This work was supported by the ICT R&D program of MSIP/IITP, Rep. of Korea (14-000-11-002, Development of Object-based Knowledge Convergence Service Platform using Image Recognition in Broadcasting Contents and the Industrial Strategic Technology Development Program (10035348, Development of a Cognitive Planning and Learning Model for Mobile Platforms) funded by the Ministry of Knowledge Economy (MKE, Korea).
+ Corresponding author.

it is essential to define the similarity between entities that reflects all information of entities. In general, an ontology entity has three kinds of information: lexical, structural, and logical information. Lexical information is expressed in labels[1] or values of some properties such as label and comment. The lexical similarity is then easily designed as a comparison of character sequences in labels or property values. The structural similarity is, however, difficult to obtain, since the structure of an entity is represented as a graph due to its various relations with other entities. Therefore, a method to compare graphs is needed to capture the structural similarity between entities. Logical information is also valuable to describe entities in an ontology. Unlike ordinary database-like knowledge repositories, some aspects of ontologies are represented with Description Logics. Thus, logical information is quite important to obtain semantic similarities between entities in the ontologies. Among three kinds of information, in this paper, we consider to handle both lexical and structural information for computing semantic similarity. Especially, we more focus on structural information in this paper.

The difficulty of comparing ontology entities with structural information can be summarized in two aspects. First, an ontology entity is complex. Thus, in most previous studies, the graph of an ontology entity has been transformed into a simpler and more manageable structure such as a vector [1], a label-sequence [2], or a pre-defined simple structure [3]. However, some structural information is lost during the transformation. For instance, Son *et al.* transformed an ontology into three types of trees: a concept tree, a property tree, and an instance tree [4]. Since the trees are transformed using only subClassOf, domain, range, and instanceOf, the information expressed with other relations are lost. The other difficulty is that the graph of an entity consists of its neighbor entities, while the neighbor entities are also graphs that have their own structural and lexical information. Therefore, not only the target entity but also neighbor entities should be essentially considered in matching entity graphs.

In order to tackle the difficulties, this paper proposes ontology kernel, a novel kernel function for ontology alignment. The ontology kernel is based on a random walk graph kernel [5] which compares graphs efficiently without explicit feature enumeration. When two graphs are given, the random walk graph kernel implicitly enumerates all possible random walks, and then the similarity between the graphs is computed using the shared random walks. As a result, it shows a good performance in comparing ordinary graphs [6, 7]. However, the nodes of an entity graph have their own structure that is represented as another graph. Therefore, the random walk graph kernel also loses the structural information of neighboring entities, unless it considers their structures additionally.

The ontology kernel, a modified random walk graph kernel solves this problem. This kernel computes the similarity of two ontology entities using not only their labels but also the structures of their neighbor entities. In addition, it considers all kinds of ontology relations in the similarity computation of entities. Therefore, the ontology kernel is robust to structural difference between the ontologies to be aligned. Especially, the kernel aligns two ontologies, even when the lexical matching of the ontologies is not trustworthy.

Evaluation of the ontology kernel is done with benchmark set and conference set from OAEI (Ontology Alignment Evaluation Initiative) 2012 campaign. The benchmark set is

[1] In this paper, the label of an entity denotes the value of the annotation property “rdfs:label”.

artificially-designed reference data, while the conference set is real data composed of the ontologies used to organize conferences. When compared with other OAEI 2012 competitors, the ontology kernel is the only method which shows high performance for both sets. This result proves the effectiveness of the ontology kernel for ontology alignment.

The rest of the paper is organized as follows. Section 2 reviews the related works on ontology alignment and graph kernel. Section 3 introduces the structure of ontology entities. Section 4 explains the random walk graph kernel, the most general graph kernel, and then Section 5 addresses the proposed ontology kernel. Section 6 shows the experimental results. Finally, Section 7 draws conclusions.

2. RELATED WORK

According to [8], the goal of ontology alignment is to generate an alignment of two given ontologies, and in this case, those generated alignments should reflect semantic relatedness of entity pairs. The alignment is composed of pairs of entities from the two ontologies that satisfy a certain relation with a certain confidence. Normally concepts and properties among various entities are aligned and the relation between entities is restricted to equivalence relation. In addition, the similarity between entities is often used instead of the confidence.

There have been proposed myriad alignment systems with their own similarity measure. WikiMatch, one of participants of the OAEI campaign, compares a pair of entities based on their lexical information [9]. Since the lexical information expressed in the entities is insufficient, WikiMatch expands the information with Wikipedia. It retrieves Wikipedia articles by querying the labels in entities. Then, the similarity between entities is replaced by that of the retrieved articles. WeSeE expands the lexical information in a similar way to WikiMatch [10]. It queries the labels to a search engine rather than to Wikipedia. These systems focus only on lexical information of entities. Thus, they ignore the semantic information lying on the entity structure, even if it is critical. In order to solve this problem, the structural information should be reflected into the alignment. However, it is not easy, since the structure of an entity is normally complex.

Various methods have been designed to reflect structural information into their similarity measure [11]. They are clustered into two categories according to the way of handling structural information. The methods that use structural similarity as an auxiliary measure belong to the first category. YAM++ [12], one of the state-of-the-art alignment systems, first aligns entities based on their lexical similarity, and then updates the alignments using the similarities of their neighbor entities to consider the entities connected with it. On the other hand, Optima+ [13] and HotMatch [14] use structure information to refine alignments. They also align entities with lexical information first. Then, they resolve the inconsistency of the alignments based on their structural information. Once structural information is used, it is helpful to performance improvement. However, it is just a supplement of lexical similarity. That is, the structural information is not used at all, if the initial alignments are consistent.

The methods that use structural information directly belong to the other category. ASMOV computes the structural similarity between entities by decomposing an entity graph into two subgraphs [15]. One subgraph expresses only ancestor-descendant rela-

tions among entities, while the other subgraph expresses all other relations given by object type properties. Thus, it loses the structural information that lies between the two subgraphs. RiMOM [2] takes a path-similarity as its structural similarity. A path of an entity is defined in RiMOM as a sequence of labels from root entity to it. Then, RiMOM uses the paths as structural information. In these methods, structural similarity is the primary similarity. Thus, even when lexical information is not sufficient, they do align ontologies. However, since their structural similarity uses just a few principal relations, some important information that is contained in other relations can be lost. For example, consider recent systems proposed by Acampora *et al.* [16, 17] or Bock and Hettenhausen [18]. In these systems, structural similarity is determined by using hierarchy distance and domain-range restriction distance. Hierarchy distance denotes the mean of the hierarchy distance from a pair of entities to their already aligned super entities. On the other hand domain-rage restriction distance is determined how many domains and ranges are shared between two properties to be aligned. Even though these studies also suggested an efficient way to combine several similarity measures and they proved effectiveness of their systems empirically, they can not avoid information loss caused by other structural aspects of ontologies like instances of concepts, real values of properties, inverse relations between properties, and so on should be ignored. Due to such structural information loss, they can miss meaningful aligns between entities in ontology alignment.

The main reason why the legacy systems do not use whole structural information is the difficulty of comparing graphs. Haussler proposed a solution to this problem, so-called convolution kernel [19]. Kernel functions have been widely applied to various applications for several decades [20, 21]. Among diverse kernels, the convolution kernel is designed to determine the similarity between structural data with their shared sub-structures. Since the structure of an ontology entity can be regarded as a graph, the similarity between entities can be obtained by a convolution kernel for a graph. For instance, the random walk graph kernel [5] can be used for ontology alignment. It regards random walks of a graph as sub-structures of the graph. Thus, the similarity of two graphs is computed by measuring how many random walks are shared by the graphs. Such graph kernels compare graphs without any structural transformation [22], but they are designed to deal with ordinary graphs whose nodes and edges are assumed to have only labels. Since the graph of an ontology entity is composed of other ontology entities, the nodes in this graph have their own structure. Therefore, the graph kernels also lose some structural information of entities when directly applied to ontology alignment.

3. ONTOLOGY ALIGNMENT

3.1 Ontology as Graph

An ontology describes concepts on a specific domain with properties and instances. Properties present relational information of concepts and they are categorized into two types: data type property and object type property. Data type properties establish relations between concepts and data types such as integer, string, and so on. On the other hand, relations among concepts are expressed with object type properties. Instances are real-world objects of concepts.

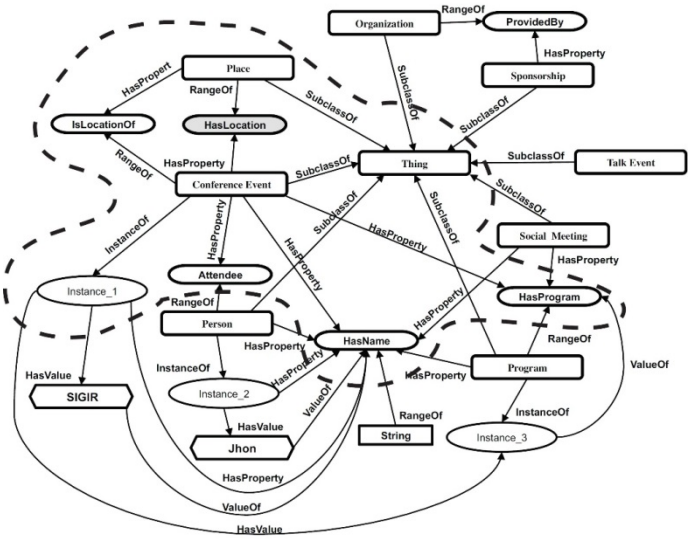


Fig. 1. An example of ontology.

An ontology is regarded as a graph of which nodes and edges are ontology entities [23], but we adopt an ontology representation that is slightly different from it. Our representation regards every entity in an ontology as a node and every axiom as an edge of the graph. Fig. 1 shows the graph structure of a simple ontology at the domain of conference organizing. This figure contains the nodes generated from five types of ontology entities: concepts, property, instances, property value types, and property values. The edges are generated from all axioms such as SubClassOf," in stanceOf, and so on. As shown in this figure, the semantics of an entity like *Conference Event* is expressed with all other related entities and relation types among them. In this paper, the graph which represents an ontology is referred as ontology graph.

Each entity of an ontology has a structure, since it has relations with other entities. Thus, it can be regarded as a subgraph of ontology graph. The subgraph for an entity is called as an entity graph. The entity graph of a specific entity can be easily obtained by extracting a subgraph from an ontology graph. It contains all nodes within a certain path length, where these nodes are the neighbor entities of the entity. All edges between nodes in the subgraph are inherited from the ontology graph. Fig. 2 depicts the entity graph of the property, HasLocation, and it is equivalent to the dotted area of Fig. 1.

3.2 Ontology Alignment with Similarity

Let E_i be a set of concepts and properties in an ontology O_i, and G_g be a graph for an entity $e \in E_i$. The alignment of two ontologies O_1 and O_2 is a list of concept-to-concept and property-to-property pairs. There are two ways to generate pairs of aligned entities. First, if one-to-one matching of entities is required, then, for each entity $e_1 \in E_1$, the best matched entity e_2^* from O_2 is determined by

$$e_2^* = \arg\max sim(G_{g_1}, G_{g_2}), \tag{1}$$

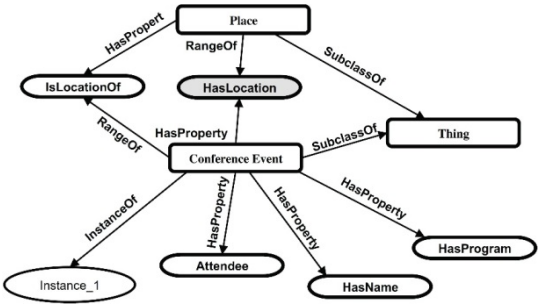


Fig. 2. An example of entity graph.

where $sim(G_{g_1}, G_{g_2})$ is the similarity between two graphs.

If one-to-many matching is assumed, many entities from O_2 can be matched to an entity in O_1. All entities in E_2 whose similarity with $e_1 \in E_1$ is larger than a pre-defined threshold θ become the matched entities of e_1. That is, for an entity $e_1 \in E_1$, a set E_2^* is matched which satisfies

$$E_2^* = \{e_2 \in E_2 \mid sim(G_{g_1}, G_{g_2}) \geq \theta\}. \tag{2}$$

Note that the key factor of Eqs. (1) and (2) is obviously the similarity, $sim(G_{g_1}, G_{g_2})$.

4. GRAPH KERNEL

The main obstacle of computing $sim(G_{g_1}, G_{g_2})$ is the graph structure of entities. One feasible solution to this problem is a graph kernel. A graph kernel maps graphs onto a feature space spanned by their subgraphs. Thus, for given two graphs G_1 and G_2, the kernel is defined as

$$K_{graph}(G_1, G_2) = \Phi(G_1) \cdot \Phi(G_2), \tag{3}$$

where $\Phi(G)$ is a mapping function which maps G onto the feature space. The kernel functions like K_{graph} can be used as a similarity, since they are the inner product of two parameter elements [24].

According to [5], it is as hard as deciding whether two graphs are isomorphic to compute any complete graph kernel with an injective mapping function for all graphs, where graph isomorphism is in NP-complete. Thus, most graph kernels focus on alternative feature representation of graphs. The random walk graph kernel, one of the most general graph kernels is a practical kernel for graphs. It uses all possible random walks as features for graphs. Let S be a set of all possible random walks. For each random walk $s \in S$ whose length is n, the corresponding feature value of a graph G is given as

$$\Phi_s(G) = \sqrt{\lambda^n} \mid \{w \in W_n(G) \mid \forall i, l(s_i) = l(w_i)\} \mid, \tag{4}$$

where λ^n is a weight for the length n and $W_n(G)$ is a set of random walks with length n in G. $l(s_i)$ and $l(w_i)$ are the ith labels in random walks s and w respectively.

With this mapping function, Eq. (3) defines the similarity between two graphs as a summation of the frequencies of all possible random walks within the graphs. Thus, all random walks should be enumerated in advance to compute the similarity. However, this is computationally infeasible. Gärtner *et al.* adopted a direct product graph as a way to avoid explicit enumeration of all random walks [5]. Let $G = (V, E)$ be a graph with a vertex set V and an edge set E. Then, the direct product graph of $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is denoted by $G_1 \times G_2 = (V_\times, E_\times)$ where $V_\times$ and $E_\times$ are its node and edge sets that are defined respectively as

$$V_\times(G_1 \times G_2) = \{(v_1, v_2) \in V_1 \times V_2 |\ l(v_1) = l(v_2)\},$$
$$E_\times(G_1 \times G_2) = \{((v_1, v'_1), (v_1, v'_2)) \in E_1 \times E_2 \mid (v_1, v'_1) \in E_1 \text{ and } (v_2, v'_2)) \in E_2 \text{ and } l(v_1, v'_1) = l(v_2, v'_2)\},$$

where $l(v)$ is the label of a node v and $l(v, v')$ is the label of an edge between two nodes v and v'. Nodes and edges are compared basically by exact string matching, but other approximate string matching methods can be also adopted.

From the adjacency matrix $A \in \Re^{|V_\times| \times |V_X|}$ of $G_1 \times G_2$, the similarity of G_1 and G_2 can be directly computed without explicit enumeration of all random walks. The adjacency matrix A has a well-known characteristic. When the adjacency matrix is multiplied n times, an element $A^n_{v_\times, v_\times}$ becomes the summation of similarities between random walks of length n from $v_\times$ to $v_\times$, where $v_\times \in V_\times$ and $v'_\times \in V_\times$ [25]. Thus, by adopting a direct product graph and its adjacency matrix, Eq. (3) is rewritten as

$$K_{graph}(G_1, G_2) = \sum_{i,j=1}^{|V_\times|} \left[\sum_{n=0}^{\infty} \lambda^n A^n \right]_{i,j}. \tag{5}$$

According to [25], the value of $K_{graph}(G_1, G_2)$ is converged as the iteration goes by. However, in general, a pre-defined value is used to set the max iteration number.

5. ALIGNMENT WITH ONTOLOGY KERNEL

5.1 Ontology Kernel

The random walk graph kernel computes the similarity between graphs without explicit feature enumeration. However, by regarding a random walk as a sequence of labels appearing on nodes and edges, its feature space spanned by random walks is not ideal for ontology graphs. In the random walk graph kernel, $s \in S$ is a label sequence of length n, which implies that all nodes and edges in graphs contain only lexical information. However, in the entity graphs, nodes contain not only lexical information but also structural information. Therefore, the feature space by S fails in reflecting all information on the entity graphs.

The ontology kernel overcomes this problem by adopting a different feature space. The feature space of the ontology kernel is spanned by S', a set of entity random walks. This set contains all random walks composed of ontology entities. That is, all nodes $s' \in S'$ are ontology entities which have both a label and a graph structure. Therefore, s' of length n is defined as

$$s' = e^1\, e^2 \ldots e^n \tag{6}$$

where $e^i = (L^i_{g'}, G^i_{g'})$ is the ith entity in s'. $L^i_{g'}$ is the label of the entity e^i and $G^i_{g'}$ is its graph structure. When the entity is a concept or a property, $G^i_{g'}$ becomes an entity graph again. Otherwise, $G^i_{g'}$ is a null graph which does not contain any node.

The ontology kernel computes the inner product of two ontology entities in the feature space spanned by S'. Let $S'_G \subset S'$ be a set of entity random walks within a graph G. The ontology kernel returns the inner product of two entity graphs G_1 and G_2 by summing all similarities between random walks $s'_{G1} \in S'_{G1}$ and $s'_{G2} \in S'_{G2}$. Assuming that the similarity between s'_{G1} and s'_{G2} is computed by a walk kernel, the ontology kernel is defined as

$$K_{graph}(G_1, G_2) = \sum_{s'_{G_1} \in S'_{G_1}} \sum_{s'_{G_2} \in S'_{G_2}} N(s'_{G_1}) N(s'_{G_2}) \cdot K_{walk}(s'_{G_1}, s'_{G_2}),$$

where $N(s'_G)$ is the frequency of the random walk s'_G appearing in G, and $K_{walk}(s'_{G1}, s'_{G2})$ is a walk kernel which returns the similarity between s'_{G1} and s'_{G2}.

One of the simple ways to define K_{walk} is to decompose a random walk into ontology entities as shown in Eq. (6). Then, the similarity between two walks is determined by production of all inner products among entities along the walks. That is, K_{walk} is defined as

$$K_{walk}(s'_{G_1}, s'_{G_2}) = \prod_{i=1}^{n} e^i_{G_1} \cdot e^i_{G_2}.$$

If $L^i_{g'}$ is independent of $G^i_{g'}$, the inner product of two entities is rewritten as

$$e^i_{G_1} \cdot e^i_{G_2} = K_l(L^i_{s'_{G_1}}, L^i_{s'_{G_2}}) + K_s(G^i_{s'_{G_1}}, G^i_{s'_{G_2}}). \tag{7}$$

Here, K_l determines the inner product of entity labels and K_s compares their graph structures. Various string matching methods such as exact matching and Levenshtein distance can be used for K_l. We use Jaro Winkler distance [26] as K_l in this paper. Since K_s computes the structural similarity of two graphs, it can be replaced by $K_{graph}(G_1, G_2)$ in Eq. (3). It is worth noting that, based on the definition of e^i, the ontology kernel should be defined recursively, however, in this paper, the kernel is defined by using only single layered architecture. That is, the graph kernel in Eq. (7) determines a similarity only with label sequences $s \in S$.

In addition, there are two facts to note about the ontology kernel. The first one is that the ontology kernel is also computed without explicit feature enumeration by adopting a direct product graph and its adjacency matrix. Thus, Eq. (5) can be used to compute the ontology kernel. The second fact is that there is a way to reduce computational cost in the ontology kernel. If we assume that only the entities with the same type can be matched like concept-to-concept, property-to-property, and instance-to-instance, the computational cost can be reduced. Under this assumption, the number of matched entities is reduced, which results in a sparse adjacency matrix. This leads to fast computation of Eq. (5), since many elements in the adjacency matrix become 0. The implementation of this assumption can be done by simply modifying Eq. (7) as

$$e^i_{G_1} \cdot e^i_{G_2} = I(e^i_{G_1}, e^i_{G_2})(K_l(L^i_{s'_{G_1}}, L^i_{s'_{G_2}}) + K_s(G^i_{s'_{G_1}}, G^i_{s'_{G_2}})), \tag{8}$$

where $I(e^i_{G_1} \cdot e^i_{G_2})$ is an indication function defined as

$$I(e^i_{G_1}, e^i_{G_2}) = 1 \text{ if } T(e^i_{G_1}) = T(e^i_{G_2}), \text{ otherwise } 0. \tag{9}$$

Here, $T(e^i_{G_1})$ returns the type of the entity $e^i_{G_1}$ such as concept, property, and instance.

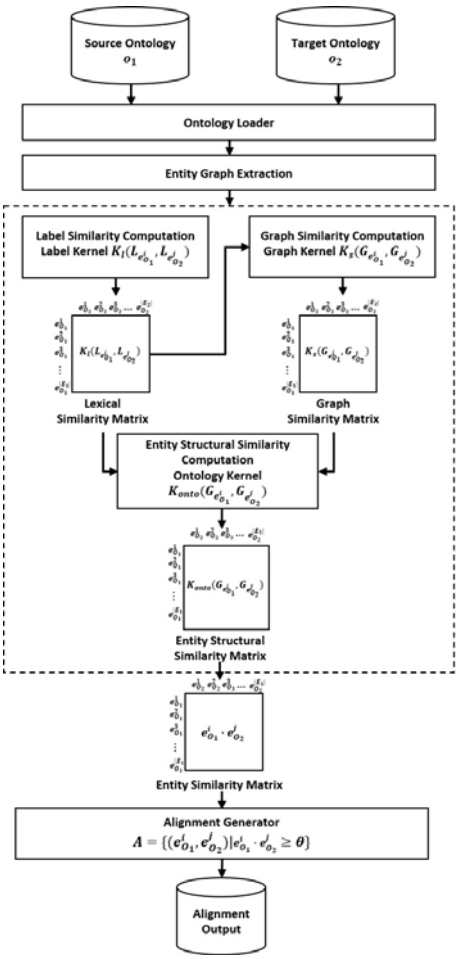


Fig. 3. Overall process to generate alignment with the ontology kernel.

5.2 System Implementation

Fig. 3 shows the overall process to obtain the alignment between two ontologies by using the ontology kernel. When two ontologies are given, graphs for all entities are extracted from the ontology graphs. Then, for each possible entity pair, lexical similarities

are measured and these similarities are stored with the lexical similarity matrix. Then, by using the lexical similarity matrix, the graph similarity matrix is constructed by the graph kernel in Eq. (5). Finally, the ontology kernel generates the entity structural similarity matrix based on both lexical similarity matrix and the graph similarity matrix. Note that all the similarity matrixes contain normalized similarities bounded from 0.0 to 1.0. In case of similarities from the graph kernel and the ontology kernel, we use a general way to normalize the kernel value. That is, similarities in a similarity matrix are measured with $K'(x,y) = K(x,y)/\sqrt{K(x,x)\cdot K(y,y)}$ to use a threshold θ in the alignment generator.

The ontology kernel is recursively defined by using the graph kernel. Thus, in the implementation of the ontology kernel, it is also adopted the kernel computation in Eq. (5), while the values in the adjacency matrix are obtained from Eq. (8). Due to such implementation, the graph similarity matrix is determined by the graph kernel in advance of the ontology kernel.

6. EXPERIMENTS

6.1 Experimental Data and Setting

Experiments are performed with the biblio in the benchmark set and the conference set prepared by Ontology Alignment Evaluation Initiative (OAEI). The biblio set is composed of one reference ontology and 47 modified versions of the reference ontology[2]. The reference ontology is numbered as 101. It has 33 named concepts, 24 object type properties, 40 data type properties, and 112 instances. The ontologies numbered as 201 and 202 are those modified from the reference ontology by changing the labels of the reference ontology randomly, but preserving its structure. On the other hand, the ontologies numbered as 221-247 are obtained by distorting the structure of the reference ontology while preserving its lexical information. The remaining 248-266 ontologies are made by changing somewhat both lexical and structural information of the reference ontology. This biblio set is designed to evaluate the performance of ontology alignment systems against diverse modifications of entities. Especially, the 248-266 ontologies are harshly modified from the reference ontology. They are difficult to align, but the performance on them is important since they reveal the functionality of alignment systems in the rigorous environment.

Both 201-202 and 248-266 ontologies also have one more distinguishing characteristic from other ontologies. These ontologies have their partial modification versions. For instance, the 250 ontology have four more ontologies, 250-2, 250-4, 250-6, and 250-8 which mean that the random string replacement of entity labels is applied for 20, 40, 60, and 80 percent of the entities respectively. However, structures of 250-2, 250-4, 250-6, and 250-8 are not modified from the 250 ontology, thus, their structures are still equivalent with the 250 ontology. These partial modified ontologies can disclose the performance change of an alignment system as the information from the labels of entities is reduced. On the other hand, these partial modified ontologies also can increase the performance of a system with rich lexical knowledge which can obtain much higher performance on less modified ontologies like xxx-2 and xxx-4. Thus, on the biblio set, we

[2] For more detail description of this data set, refer to http://oaei.ontologymatching.org/2012/benchmarks/index.html.

present both performances of a system with and without partial modified ontologies.

The conference set consists of seven ontologies describing organizing conferences and 21 reference alignments among them. A simple statistics on these ontologies is given in Table 1. As shown in this table, the ontologies have only concepts and properties. The average number of concepts is 72, and that of properties is 44.43. Even if the ontologies in this set come from the same domain, they are designed independently. As a result, the entity expressions in each ontology are slightly but naturally different one another. Therefore, the alignment results with this set show the real-world performance of alignment systems[3]. To measure performances on the conference set, the *ra1* alignment is used as a gold standard alignment.

The proposed method, ontology kernel is compared with other ontology alignment systems that participated in the OAEI 2012 campaign [31]. All parameters of the proposed method are set heuristically. The maximum length of random walks is set to be two. In case of λ in Eq. (4), it firstly set as 0.70, and as the iteration goes by, it is exponentially decreased as $\lambda = 0.70^n$ for the nth iteration. To handle ontologies with randomly modified labels like 201 and 202 ontologies in the biblio set, the ontology kernel is adopted a simple heuristic process: (1) determine the average lexical similarity between two ontologies; (2) when the average lexical similarity is not exceeded the pre-defined threshold (in experiments, we used 0.8 in this step), labels of all entities in both ontologies are changed with their entity type; (3) an indicator function which returns 1.0 when two entities have the same type, or 0.0 is adopted as the label kernel in Eq. (8). The types of entities used in the second step are {*class*, *property*, *individual*}. In case of the values, we used primitive types of values such as {*string*, *int*, *float*, …}. The performance of ontology alignment is measured by harmonic mean of precisions, recalls, and micro average F-measures, since they are standard in the OAEI campaign.

Table 1. Simple statistics of the conference set.

Ontology	No. of concepts	No. of properties
Cmt	36	59
Sofsem	60	64
ConfTool	38	36
Edas	104	50
Ekaw	77	33
Iasted	140	41
Sigkdd	49	28
Average	72.00	44.43

6.2 Experimental Results

Table 2 shows the performance of the ontology kernel against other competitors with the biblio set. OntoK in this table is the proposed ontology kernel, while ‘edna’ is a simple edit distance and is regarded as a baseline. In this table, performances of all systems are measured with F-measure and, the performances without partial modified on-

[3] The conference set is imported from SEALS platform directly with Conference testsuite-ID (http://oaei.ontologymatching.org/2012/conference/index.html).

Table 2. The performance of alignment systems on the *biblio* set in F-measure.

	YAM++	MapSSS	AROMA	WeSeE	HotMatch
101	1.00	1.00	1.00	0.95	0.83
201, 202	0.92 (0.86)	0.94 (0.91)	0.89 (0.81)	0.78 (0.60)	0.77 (0.62)
221-247	1.00	1.00	1.00	0.98	0.92
248-266	0.76 (0.35)	0.81 (0.61)	0.67 (0.29)	0.56 (0.00)	0.54 (0.01)
Average	0.83 (0.79)	0.87 (0.86)	0.77 (0.77)	0.69 (0.68)	0.66 (0.64)
	Hertuda	WikiMatch	Optima	ASE	AUTOMSv2
101	0.94	0.91	1.00	1.00	0.94
201, 202	0.78 (0.62)	0.73 (0.56)	0.75 (0.60)	0.57 (0.45)	0.79 (0.62)
221-247	0.94	0.89	0.96	0.78	0.98
248-266	0.55 (0.00)	0.50 (0.00)	0.48 (0.00)	0.41 (0.00)	0.56 (0.00)
Average	0.68 (0.67)	0.62 (0.62)	0.63 (0.67)	0.51 (0.53)	0.69 (0.69)
	GOMMA	LogMap	LogMapLt	MaasMatch	MEDLEY
101	0.91	0.90	0.91	1.00	0.84
201, 202	0.80 (0.69)	0.49 (0.00)	0.51 (0.00)	0.52 (0.09)	0.45 (0.00)
221-247	0.87	0.86	0.88	0.97	0.86
248-266	0.57 (0.16)	0.46 (0.00)	0.49 (0.00)	0.46 (0.05)	0.46 (0.00)
Average	0.67 (0.65)	0.56 (0.56)	0.59 (0.58)	0.56 (0.52)	0.54 (0.55)
	ServOMap	ServOMapLt	edna	**OntoK**	
101	0.99	0.44	0.78	**1.00**	
201, 202	0.56 (0.00)	0.21 (0.00)	0.31 (0.01)	**0.94 (0.90)**	
221-247	0.87	0.59	0.82	**1.00**	
248-266	0.48 (0.00)	0.28 (0.00)	0.33 (0.01)	**0.78 (0.56)**	
Average	0.58 (0.58)	0.33 (0.34)	0.41 (0.42)	**0.84 (0.81)**	

tologies are also represented by using parenthesis. As shown in Table 2, all competitors and the ontology kernel outperform 'edna' except ServOMapLt that is a restricted version of ServOMap. Thus, if we consider those restricted versions of particular systems like ServOMapLt and LogMapLt, then all competitors of OAEI 2012 achieved better performances than the performance of edna, 0.41. The ontology kernel achieves the second best performance with 0.81 of average F-measure. The only system that outperforms ontology kernel is MapSSS with 0.86 of average F-measure. However, MapSSS depends on the lexical information heavily and uses only a few specific relations. Thus, when the ontologies to be aligned contain many other relations, it is hard to reflect structural information in alignments. Actually, it is one of the worst systems for the conference set.

The alignment of the reference ontology with the 101 ontology is a self-alignment, since 101 ontology is the reference ontology itself. Thus, F-score of 1.00 is expected for all systems. However, surprisingly, among 19 systems, 12 systems fail in achieving the complete alignment. They are the systems that expand the lexical information of entities with an external resource such as Wikipedia or Bing, a web search engine. Their poor performance implies that this kind of expansion works as a noise.

The ontologies numbered as 201 and 202 are those of which labels are changed with a random string. Thus, the systems that use only the lexical information show low performance in the alignment with these ontologies. WeSeE and Wiki-Match are such alignment systems. One of interesting results is that most systems showed large gaps between performance with and without partial modified ontologies. However, OntoK

showed the performance difference within 0.05. One of efficient way to achieve such a small difference is to use structural information, because the effect of lexical information for the alignments is diminished in 201-202 ontologies. Thus, this small difference demonstrates the superiority of OntoK on handling structural information.

The ontologies numbered as 221-247 are designed to see the ability of each system to cope with the structural difference between ontologies. Thus, only the systems that utilize the structural information are expected to achieve good performance. However, contrary to our expectation, most systems show F-measure of over 0.90. Especially, some systems including the ontology kernel achieve F-measure of 1.00 for them. This is because these ontologies still possess the lexical information enough for the alignment, even though their structures are a little bit distorted.

The 248-266 ontologies are most difficult to align, since they are generated by modifying both lexical and structural information of the reference ontology. Thus, the systems have to consider the lexical and structural difference of the ontologies at the same time for the alignment of these ontologies. However, most systems consider just one of them or process the lexical information first. As a result, their performances are extremely low when lexical information is not available. On the other hand, the ontology kernel achieves F-measure of 0.78 (0.56) for them, while F-measures of many other systems are lower than 0.30. This superiority of the ontology kernel comes from its simultaneous consideration of the lexical and structural information.

Table 3 compares the performances of the alignment systems on the conference set. The ontology kernel is located on the fifth best position in this set. Unfortunately, the performances of many other systems are lower than or equal to that of 'baseline2', the baseline method which showed F-measure of 0.59. Especially, MapSSS and AROMA

Table 3. The performance of alignment systems on the *conference* set.

	YAM++	MapSSS	AROMA	WeSeE	HotMatch
Prec.	0.81	0.50	0.33	0.76	0.71
Rec.	0.69	0.51	0.48	0.49	0.51
F-measure	0.75	0.50	0.39	0.60	0.59
	Hertuda	WikiMatch	Optima	ASE	AUTOMSv2
Prec.	0.74	0.74	0.62	0.63	0.67
Rec.	0.50	0.50	0.68	0.43	0.36
F-measure	0.60	0.60	0.65	0.51	0.47
	GOMMA	LogMap	LogMapLt	CODI	MEDLEY
Prec.	0.85	0.82	0.73	0.74	0.54
Rec.	0.47	0.58	0.50	0.57	0.50
F-measure	0.61	0.68	0.59	0.64	0.52
	ServOMap	ServO-MapLt	MaasMatch	Baseline2[4]	**OntoK**
Prec.	0.73	0.88	0.63	0.79	**0.81**
Rec.	0.46	0.40	0.57	0.47	**0.52**
F-measure	0.56	0.55	0.60	0.59	**0.63**

[4] In OAEI 2012, there exist two baselines. Baseline1 denote the system with an exact matching, while baseline2 is one with the edit-distance based matching. Due to the lack of space, this paper only contains the performance of baseline2, since baseline2 showed better performance than baseline1.

achieve lower performances than 'edna', even though their performances are very high for the biblio set. On the other hand, systems like LogMap, CODI, and Optima showed better performance than the ontology kernel in the conference set, while their performances on the biblio set is worse than the ontology kernel. Onty the ontology kernel and YAM++ achieve much high performances in both biblio and conference sets. Especially, YAM++ shows the best performance for the conference set. However, note that YAM++ is worse than the ontology kernel for the biblio set. As explained in Section 2, YAM++ uses the structural information of entities as a supplement of the lexical information. Thus, its performance gets low, when two ontologies to be aligned are much different structurally. This is why it shows lower performance than the ontology kernel for 248-266 ontologies of the biblio set.

On the other hand, the ontology kernel is not much affected from the structural difference of the ontologies. Rather, it manages the difference effectively, since it utilizes all kinds of relations and structural information of entities. From the fact that the ontology kernel achieves high performance consistently for both sets, its high generality is proved. That is, the ontology kernel performs well regardless of the extent of ontology difference.

6.3 Discussion for the Time Complexity of the Ontology Kernel

The ontology kernel showed its superiority through experiments with both the biblio and conference sets. Even though its performances are always highly ranked among various alignment systems, there still exists a room to enhance the ontology kernel with respect to computational cost. Based on the graph kernel, the ontology kernel is designed to reflect the structural characteristic of the ontology. In this case, the time complexity of the ontology kernel, thus, is strongly governed by the complexity of the graph kernel. Basically, the time complexity of the graph kernel is $O(n^6)$ and it can be reduced up to $O(n^3)$ [27], where n is the number of nodes in two compared graphs. As a result, the ontology kernel that is the nested implementation of the graph kernel as shown in Fig. 3 have the time complexity of $O(2n^3)$.

In this paper, by using the entity type comparison in Eq. (9), the time complexity of the graph kernel can be reduced up to $O(n_c^3 + n_p^3 + n_i^3)$, where n_c, n_p, and n_i are the numbers of concepts, properties, and instances in two ontologies respectively and $n = n_c + n_p + n_i$. In case of the ontology kernel, it has the time complexity of $O(2 \times (n_c^3 + n_p^3 + n_i^3))$, since the ontology kernel is implemented by nesting the graph kernel twice. Theoretically, $n^3 \geq 2 \times (n_c^3 + n_p^3 + n_i^3)$ is not guaranteed. However, when an ontology contains similar numbers of concepts, properties, or instances, the ontology kernel should be faster than the graph kernel.

Fig. 4 shows actual computation time of both the ontology kernel and the graph kernel on the conference set. As shown in this figure, the ontology kernel averagely spends half of computation time comparing with the graph kernel. However, the time complexity of the ontology kernel is still much high and it can be an obstacle to align huge ontologies. Thus, further improvement is demanded to reduce the time complexity of the ontology kernel and it is our future work. Fortunately, recent work on the graph kernel suggests elegant ways to overcome this problem by using other sub-structure of graphs [28, 29] or by adopting efficient data structures [30]. We will apply such technical advances in the ontology kernel.

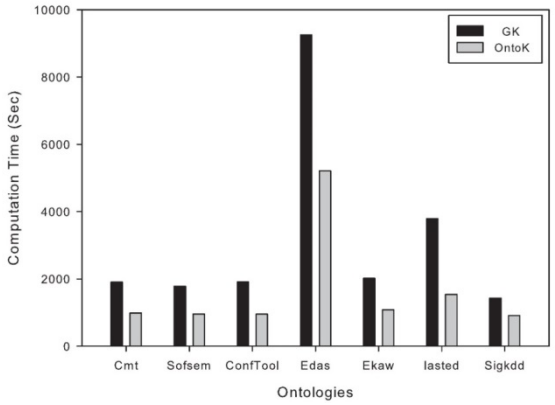


Fig. 4. The computation time of the graph kernel and ontology kernel.

7. CONCLUSIONS

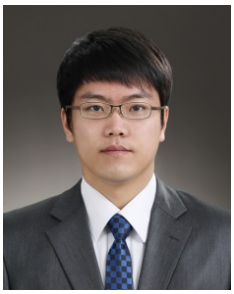
This paper proposed a novel graph kernel, the ontology kernel, to align ontologies from the same domain. Since all entities in an ontology such as concepts and properties are represented as graphs, the ontology kernel aligns the ontologies by comparing their entity graphs. For accurate alignment of ontologies, it is essential to use both lexical and structural information of the entity graphs in computing the entity similarity. One thing to note especially for the use of structural information is that the neighbor nodes of an entity graph are also ontology entities and thus they have their own structure. In order to reflect this structural characteristic of entity graphs, the ontology kernel projects the entity graphs onto the space spanned by entity random walks, the sequences of ontology entities. Then, the similarity between entities is computed using shared random walks by the entities. Since each of these entity random walks has its own lexical and structural information, both kinds of information is considered simultaneously in the ontology kernel. As a result, the ontology kernel showed high performance for two data sets used in OAEI 2012 campaign. In addition, the ontology kernel utilizes all kinds of relations, not just a few dominant relations. That is, it does not lose any information lying on the entity graphs. Therefore, it aligns ontologies well even when their structure or lexical information is much unalike. Its superiority over other OAEI competitors in 248–246 ontologies of the biblio set and the conference set prove its generality against the extent of ontology difference.

REFERENCES

1. J. Madhavan, P. Bernstein, and E. Rahm, "Generic schema matching with Cupid," in *Proceedings of the 27th International Conference on Very Large Data Bases*, 2001, pp. 49-58.
2. J. Tang, J. Li, B. Liang, X. Huang, Y. Li, and K. Wang, "Using Bayesian decision for ontology alignment," *Journal of Web Semantics*, Vol. 4, 2006, pp. 243-262.
3. P. Mitra, N. Noy, and A. Jaiswal, "OMEN: A probabilistic ontology mapping tool,"

in *Proceedings of the 4th International Semantic Web Conference*, 2005, pp. 537-547.
4. J. Son, S. Park, and S. Park, "An ontology alignment based on parse tree kernel for combining structural and semantic information without explicit enumeration of features," in *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence*, 2008, pp. 468-474.
5. T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Proceedings of the 16th International Conference on Learning Theory*, 2003, pp. 129-143.
6. U. Lösch, S. Bloehdorn, and A. Rettinger, "Graph kernels for RDF data," in *Proceedings of the 9th Extended Semantic Web Conference*, 2012, pp. 134-148.
7. K. Borgwardt, C. Ong, S. Schönauer, S. Vishwanathan, A. Smola, and H. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, Vol. 21, 2005, pp. i47-i56.
8. J. Euzenat, "Semantic precision and recall for ontology alignment evaluation," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007, pp. 348-353.
9. S. Hertling and H. Paulheim, "WikiMatch – using Wikipedia for ontology matching," in *Proceedings of the 7th International Workshop on Ontology Matching*, 2012, pp. 37-48.
10. H. Paulheim, "WeSeE – match results for OEAI 2012," in *Proceedings of the 7th International Workshop on Ontology Matching*, 2012, pp. 213-219.
11. J. Euzenat and P. Shvaiko, *Ontology Matching*, Springer-Verlag, Berlin Heidelberg, 2007.
12. D. Ngo and Z. Bellahsene, "YAM++ – results for OAEI 2012," in *Proceedings of the 7th International Workshop on Ontology Matching*, 2012, pp. 266-233.
13. U. Thayasivam, T. Chaudhari, and P. Doshi, "Optima+ results for OAEI 2012," in *Proceedings of the 7th International Workshop on Ontology Matching*, 2012, pp. 181-188.
14. T. T. Dang, A. Gabriel, S. Hertling, P. Roskosch, M. Wlotzka, J. R. Zilke, F. Janssen, and H. Paulheim, "Hotmatch results for OAEI 2012," in *Proceedings of the 7th International Workshop on Ontology Matching*, 2012, pp. 145-151.
15. T. Jean-Mary, E. Shironoshita, and M. Kabuka, "Ontology matching with semantic verification," *Journal of Web Semantics*, Vol. 7, 2009, pp. 235-251.
16. G. Acampora, V. Loia, S. Salerno, and A. Vitiello, "A hybrid evolutionary approach for solving the ontology alignment problem," *International Journal of Intelligent Systems*, 2012, Vol. 27, pp. 189-216.
17. G. Acampora, V. Loia, and A. Vitiello, "Enhancing ontology alignment through a memetic aggregation of similarity measures," *Information Sciences*, Vol. 250, 2013, pp. 1-20.
18. J. Bock and J. Hettenhause, "Discrete particle swarm optimisation for ontology alignment," *Information Sciences*, Vol. 192, 2012, pp. 152-173.
19. D. Haussler, "Convolution kernels on discrete structures," Technical Report, No. UCS-CRL-99-10, UC Santa Cruz, 1999.
20. S. Parsa and S. Naree, "A new semantic kernel function for online anomaly detection of software," *ETRI Journal*, Vol. 34, 2012, pp. 288-291.

21. M. Ahmed, J. Kim, R. Mao, J. Song, and H. Li, "Distributed channel allocation using kernel density estimation in cognitive radio networks," *ETRI Journal*, Vol. 34, 2012, pp. 771-774.
22. F. Costa and K. Grave, "Fast neighborhood subgraph pairwise distance kernel," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 255-262.
23. P. Shvaiko and J. Euzenat, "A survey of schema-based matching approaches," *Journal on Data Semantics*, 2005, pp. 146-171.
24. N. Srebro, "How good is a kernel when used as a similarity measure?" in *Proceedings of the 20th Annual Conference on Computational Learning Theory*, 2007, pp. 323-335.
25. A. Airola, S. Pyysalo, J. Björne, T. Pahikkala, F. Ginter, and T. Salakoski, "All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning," *BMC Bioinformatics*, Vol. 9, 2008.
26. M. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of tampa," *Journal of the American Statistical Association*, Vol. 84, 1989, pp. 414-420.
27. S. V. N. Vishwanathan, N. Schraudolph, R. Kondor, and K. Borgwardt, "Graph kernels," *Journal of Machine Learning Research*, Vol. 11, 2010, pp. 1201-1242.
28. F. Costa and K. Grave, "Fast neighborhood subgraph pairwise distance kernel," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 255-262.
29. Y. Zhang, H. Lin, Z. Yang, and Y. Li, "Neighborhood hash graph kernel for protein-protein interaction extraction," *Journal of Biomedical Informatics*, Vol. 44, 2011, pp. 1086-1092.
30. S. Hido and H. Kashima, "A linear-time graph kernel," in *Proceedings of the 9th IEEE International Conference on Data Mining*, 2009, pp. 179-188.
31. J. L. Aguirre, K. Eckert, J. Euzenat, A. Ferrara, W. R. v. Hage, L. Hollink, C. Meilicke, A. Nikolov, D. Ritze, F. Scharffe, P. Shvaiko, O. Šváb-Zamazal, C. Trojahn, E. Jiménez-Ruiz, B. C. Grau, and B. Zapilko, "Results of the ontology alignment evaluation initiative 2012," in *Proceedings of the 7th ISWC Workshop on Ontology Matching*, 2012, pp. 73-115.



Jeong-Woo Son received his MS and Ph.D. degrees in Computer Science from Kyungpook National University, Korea in 2007 and 2012 respectively. Now, he is a Researcher in Electronics and Telecommunications Research Institute, Korea. His research focuses on machine learning, natural language processing, information retrieval and semantic web.

Hee-Guen Yoon is a Ph.D. candidate in Kyungpook National University, Korea. He received his MS in 2009. His main research interests include machine learning and natural language processsing.

Seong-Bae Park received his MS and Ph.D. degree in Computer Science from Seoul National University, Korea in 1996 and 2006 respectively. Now, he is an Associated Professor in the School of Computer Science and Engineering, Kyungpook National University. He focuses on machine learning, natural language processing, text mining, and bioinformatics.