JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 34, 519-534 (2018)
DOI: 10.6688/JISE.201803_34(2).0013

An Efficient Matching Algorithm for Fuzzy RDF Graph

GUAN-FENG LI[1] AND ZONG-MIN MA[2,3,+,*]
[1]*College of Computer Science and Engineering*
Northeastern University
Shenyang, 110819 P.R. China
[2]*College of Computer Science and Technology*
Nanjing University of Aeronautics and Astronautics
Nanjing, 211106 P.R. China
[3]*Collaborative Innovation Center of Novel Software Technology and Industrialization*
Nanjing, 210023 P.R. China

The rapid growth of RDF (Resource Description Framework) data shows a steady trendy of decentralization. Identify correspondences among these data sources is an important task. Concentrating on the RDF data with fuzzy information, in this paper, we propose a unified framework combined multiple measures of similarity for automatically solving the problem of fuzzy RDF data matching. For this purpose, syntactic and semantic similarities are calculated based on element labels of RDF graph. Motivated by the structural characteristics of fuzzy RDF graph, we introduce an effective similarity measure approach by interactively utilizing structural information, in which we take fuzzy edge values and edge similarities into consideration. It is proved that the iterative computation of the proposed similarity approach converges. Finally, we combine these results of the individual metrics and extract the correspondences based on the total similarities. The experimental results show that our method can effectively identify correspondences among the fuzzy RDF graphs.

Keywords: fuzzy RDF, graph matching, similarity, iterative computation, label similarity, structural similarity

1. INTRODUCTION

Data matching is the process of bringing data from different data sources together and comparing them to find out whether they represent the same real-world object in a given domain [4]. Data matching processes can be applied to many data management research areas, such as a query or search over web data sources, integration of heterogeneous data sources, and so on [22, 29]. In the context of the Semantic Web, there has been a growing interest in using RDF data model, a recommendation by the World Wide Web Consortium [8] for describing resources on the Web. The increasing amount of RDF data that are captured in different information stores has resulted in a strong need for automatically discovering correspondences among heterogeneous data source [6, 14].

RDF data have a natural representation in the form of labelefLd directed graphs, in which vertices represent resources and values (also called literals), and edges represent semantic relationships between resources. So, RDF data matching problem has been often addressed in terms of graph matching approach. Unfortunately, it has been proved that the complexity of the traditional graph matching algorithm based on isomorphic

Received October 13, 2016; revised December 19, 2016; accepted January 11, 2017.
Communicated by Shyi-Ming Chen.
[*] This work was supported by the National Natural Science Foundation of China (61370075 and 61772269).
[+] Corresponding author: zongminma@nuaa.edu.cn

graph is NP-complete [18]. For this reason, the approaches to approximate matching on graph data mainly rely on similarity metrics to reduce the problem complexity. Because of the independent encoding system in different data sources and the insufficient linguistic information, directly applying existing linguistic-based techniques [2, 26] to compare element (*e.g.*, a vertex or edge) labels may obtain false mapping. Furthermore, these approaches ignore the fact that a part of the vertices are *blank vertices* in RDF graph. Structural similarity should be considered in addition to the classic typographical similarity. The basic idea is to employ the topological properties of RDF graph for obtaining the correspondences pairs. There is a kind of structural similarity approach [16], which is particularly suitable for the lack of syntactical information (*blank vertices* in our case) in graphs. Note that this kind of approaches concerns only the pure structural similarity, while the utilization of other evaluation methods such as string similarities is not considered. Furthermore, it does not take edges similarities into consideration, which are key properties in RDF graph.

In the real-world applications, information is usually vague or ambiguous [34]. Consequently, several recent efforts have devoted to develop fuzzy RDF model [20, 25]. However, these proposals lack enough expressive power to properly describe multiple granularity of data fuzziness in RDF. We argue that the semantic of an RDF graph relies on the connectivity of the resources described. This is not properly taken into account in a mere triple vagueness. Hence, we propose a fuzzy RDF graph model that combines two common types of fuzziness based on fuzzy graph theory [17]. Specifically, we consider:

(1) Fuzziness about the label values of vertices (*i.e.*, vertices label value fuzziness).
(2) Fuzziness about whether particular edges exist (*i.e.*, edge existence fuzziness).

Fuzzy RDF data matching is a fundamental problem in the integration of fuzzy RDF data. Based on the fuzzy RDF data model, we propose an approach for fuzzy RDF graph matching in this paper. The method computes multiple measures of similarity among graph elements: syntactic, semantic and structural. These measures are composed in a principled manner for graph matching. In particular, an iterative similarity function is introduced with the consideration of structural information of fuzzy RDF graph. The main contributions of this paper are summarized as follows:

(1) We introduce a general fuzzy RDF graph model that can capture fuzziness in the vertex and edge. We formalize the problem of fuzzy RDF graph and graph matching.
(2) We propose a graph matching approach based on a unified framework to generate graph alignments over the proposed fuzzy RDF graph model.
(3) We define the structural similarity function based on the idea of propagating similarity, which is computed by incorporating edge similarity criteria in the iteration.
(4) We evaluate our approach on synthetic datasets. It is shown that our approach can effectively distinguish the differences of size and content among the RDF graphs.

The rest of this paper is organized as follows. In Section 2, we give a brief overview of related work. In Section 3, we introduce some preliminary notions and definitions. In Section 4, we propose a unified framework that combines multiple measures of computing similarity between fuzzy RDF graphs. We present the experimental evaluation of our approach in Section 5. In Section 6, we draw conclusions and sketch some future work.

2. RELATED WORK

The present work in this paper is closely related to the issues of uncertain RDF modeling and classical RDF matching.

Since RDF data may contain inconsistent or imprecise information in the real-world applications, it is important to investigate the combination of RDF and uncertainty.

First, probability theory has recently been applied to represent and manipulate uncertain RDF data. Fukushige [30] extended the RDF vocabulary to allow probabilistic confidence existing in RDF data. Huang *et al.* [11] modeled uncertainty RDF data by a probabilistic database. Their model assumes that RDF triples have independent existence probabilities to appear in reality. In contrast, Lian *et al.* [28] proposed a different model of probabilistic RDF graphs, which has correlated and uncertain vertex labels rather than uncertain edges. Moustafa *et al.* [27] propose a general probabilistic graph model that combines attribute value uncertainty, edge existence uncertainty and identity uncertainty three common types of uncertainty.

Second, fuzzy set theory has recently been used for representing fuzzy RDF data. In [20], a syntactic and semantic extension of RDF was proposed to represent fuzzy data. The syntax is extended from a triple to a quadruple by adding a fuzzy value to the triple. Straccia [25] presented a minimal deductive system for fuzzy RDF in a general setting, in which the triples were annotated with a degree of truth in [0, 1]. More recently, a work somewhat close to ours was [21] where authors established fuzzy RDF graph data model by using fuzzy graph theory. A novel fuzzy RDF data model was proposed in [32], which took the fuzziness in values of the triple's items into consideration. This work focused on how to represent fuzzy RDF data rather than efficient matching of fuzzy RDF data.

There are several efforts in RDF data matching. Basically, we can identify three types of approaches: schema-based mapping, instance-based matching and structure-based methods. Schema-based methods [3, 12] rely on alignments or mapping rules to identify the same real-world objects. Instance-based methods [22] seek to identify the instances of a real-world object in two different datasets. Structure-based methods usually employ the topological properties of a RDF graph. Carroll [14] proposed a RDF graph matching algorithm based on vertices of classification and graph isomorphism. Zhang *et al.* [6] proposed a similarity-oriented RDF graph matching approach for ranking linked data, which considers the element-level and structure-level similarity of statements. Raimond *et al.* [33] proposed an algorithm that considers both the similarity of the resources and the similarity of their neighbors for matching music-related datasets. Matching evidence was propagated through other resources via object properties, in a similar manner to the similarity flooding algorithm proposed in [16].

The matching approach proposed in the paper combines the semantic similarities of fuzzy RDF label and structural heuristics to generate the overall similarities. As far as the semantic similarity measure of fuzzy RDF data, one of the most useful approaches is Granular Computing [10], which allows for a high level of man-machine cooperation by providing a framework in which concepts can be modeled in a manner amenable to both. Currently granular computing has been widely investigated in the literature [3, 9].

3. FUZZY RDF GRAPH MODEL AND MATCHING SEMANTICS

Let U be a set of URI references, L a set of literals and P a set of properties, an RDF triple is a triple (s, p, o) in $U\times P\times(U\cup L)$. The *element* of the RDF triples s, p, and o is called the subject, property (also called predicate) and object respectively. A finite set of RDF triples G_D is called a RDF graph in which every triple (s, p, o) describes a directed edge labeled with p from the vertex labeled with s to the vertex labeled with o.

Definition 1 (*Fuzzy RDF triple*). A triple (μ_s/s, μ_p/p, μ_o/o) is called a fuzzy RDF triple, where s is fuzzy subject and $\mu_s\in[0, 1]$ denote the membership degree of subject to the universe of a RDF graph, p is a fuzzy predicate and $\mu_p\in[0, 1]$ express the fuzzy degree to the property or relationship being described, o is fuzzy object and $\mu_o\in[0, 1]$ represent the fuzzy degree of the value for the property or relationship.

Definition 2 (*Fuzzy RDF data graph*). Fuzzy RDF data graph G is represented by a 6-tuple (V, E, Σ, L, μ, ρ) where V is a finitude set of vertices, $E\subset V_i\times V_j$ is a set of directed edges, Σ is a set of labels, L: $V\cup E\rightarrow\Sigma$ is a function assigning labels to vertices and edges respectively, μ: $V\rightarrow(0, 1]$ is a fuzzy subset, and ρ: $E\rightarrow(0, 1]$ is a fuzzy relation of on fuzzy subset μ. Note that $\rho(v_i, v_j)\leq\mu(v_i)\wedge\mu(v_j)$ $v_i, v_j\in V$.

In Definition 2, each vertex $v_i\in V$ of graph G has one label, $L(v_i)$, corresponding to either *subjects* or *objects* in RDF triples. Moreover, $(v_i, v_j)\in E$ is a directed edge from vertex v_i to v_j, with an edge label $L(v_i, v_j)$ that corresponds to the *predicate* in RDF triples. The label of vertex is associated with a membership degree indicating the possibility that vertex take the label, and the fuzzy value associated with each edge represents the amount of disagreement on the corresponding relationship between vertices. Note that a crisp RDF graph is simply a special case of fuzzy RDF data graph (where μ: $V\rightarrow\{0, 1\}$ for all $v_i\in V$ and ρ: $V\times V\rightarrow\{0, 1\}$ for all $(v_i, v_j)\in E$).

Definition 3 (*Predecessor set and successor set*). Let G be a fuzzy RDF graph, for any vertex $v\in V$ of graph G, $pre(v) = \{v'|(v, v')\in E\}$ is the predecessor set (*i.e.*, forward neighbors) of v and $succ(v) = \{v'|(v', v)\in E\}$ is the successor set (*i.e.*, backward neighbors) of v.

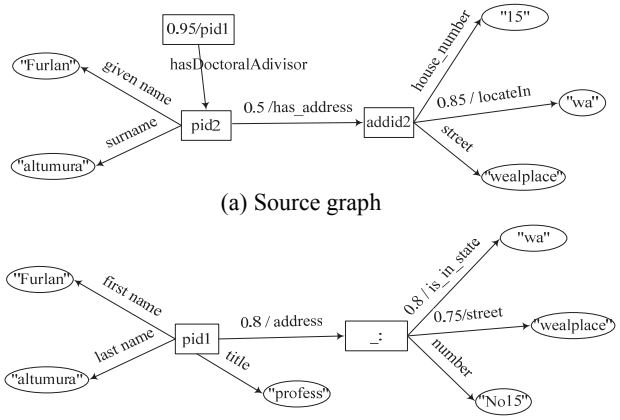


Fig. 1. The fuzzy RDF graphs (a) Source graph; (b) Target graph.

For example, Figs. 1 (a) and (b) illustrate two fragments of fuzzy RDF data graph with some fuzzy elements, and crisp ones. The edge "pid2-has_address-addid2" associated with membership degree 0.5 in target graph Fig. 1 (a) represents the fact that the person labeled pid2, whose address label is addid2. And the possibility of the fact is 0.5. Note that opaque labels exist as shown in Fig. 1 (b). The resource "_:" is distinct from others, and it makes the resource name garbled. According to the RDF specification [7], a blank vertex can be assigned an identifier prefixed with "_:".

With the presence of dislocated matching [29], some vertices in a RDF graph can be starting/ending vertices. We add the following restriction on fuzzy RDF graph:

(1) There is one and only one vertex in the RDF graph that is called the *home vertex*, denoted by $\hat{v}$, which indicates the virtual beginning/end of all paths in a RDF graph. We specify that the label of the *home vertex* is "_: H", *i.e.*, $L(\hat{v})$ = "_:H".
(2) There are paths from the *home vertex* to any other vertices in fuzzy RDF graph. That is, for each vertex $v \in V$ except $\hat{v}$, we add two edges $(v, \hat{v})$ and $(\hat{v}, v)$. Thus, a path can begin or end with vertex v at all the locations where v occurs. Moreover, we associate $\rho(v, \hat{v}) = \rho(\hat{v}, v) = \mu(v)$, *i.e.*, we regard the fuzzy degree associated with each vertex represents the possibility that the vertex exists in the graph as the fuzzy degree of edge from home vertex to the vertex.

Definition 4 (*Fuzzy RDF graph matching*). Given two fuzzy RDF graphs G_S and G_T from a given domain, the matching problem is to identify all correspondences between graphs G_S and G_T representing the same real-world object. The match result is typically represented by a set of correspondences, sometimes called a mapping. A correspondence $c = (id, E_s, E_t, m)$ interrelates two element E_s and E_t from graphs G_S and G_T. An optional similarity degree $m \in [0, 1]$ indicates the similarity or strength of the correspondence between the two elements.

Definition 5 (*Similarity function*). Let G_S and G_T are two datasets, a similarity function is defined as: $F_s(s, t) \rightarrow [0, 1]$, where $(s, t) \in G_S \times G_T$, *i.e.*, the function computers a normalized value for every pair (s, t). The higher the score value, the more similar *s*and *t*are. The advantage of using similarity functions is to deal with a finite interval for the score values.

4. FUZZY RDF GRAPH MATCHING APPROACH

4.1 Framework

Matching procedure takes as input two fuzzy RDF graphs and outputs a set of correspondences of the two graphs. Fig. 2 illustrates an overview of the framework and it has three main stages: First, the procedure is to compute a vertex-to-vertex similarity score using different similarity functions. Label similarity functions adopt different computation strategies to compute multiple types of vertex label similarity scores. Structural similarity function iteratively computes similarity scores for every vertex pair by aggregating the similarity scores of edge and immediate neighbors' vertices. Then, we obtain the overall similarity by combining label similarity scores and structural similarity

scores. Finally, we select the potential correspondences based on the similarity scores and include them in the alignment.

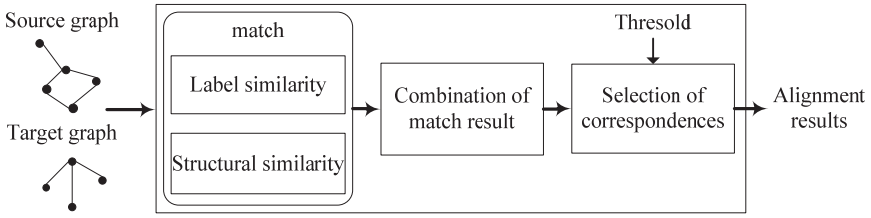


Fig. 2. The framework of fuzzy RDF graph matching.

4.2 Label Similarity Function

4.2.1 Syntactic similarity

Intuitively, the element label denoting an element typically captures the most distinctive characteristic of the element in the RDF graph model. The syntactic similarity assigns a normalized similarity value to every pair (s, t) by applying the Levenshtein distance [26] to the name labels of s and t. Formally the syntactic similarity $sim_{sy}(s, t)$ between two name labels s and t is defined as follows.

$$sim_{sy}(s,t) = 1 - \frac{LD(s.label, t.label)}{\max(|s.label|, |t.label|)}. \quad (1)$$

Here *s.label* and *t.label* denote the name label of s and t, respectively, max(|*s.label*|, |*t.label*|) is the max length of the name string in s and t, and $LD(w_1, w_2)$ is the Levenshtein distance between two words w_1 and w_2.

4.2.2 Semantic similarity

For semantic similarity, we use WordNet::Similarity package [24] to get the semantic relatedness between element labels based on their linguistic correlations. In our work, we use Jaccard similarity [2] measures. In many case, the element labels whose relatedness is being measure are phrase or short sentence, *e.g.*, "house-number" and "room number" in Fig. 1. In these cases, we need a new measure that computes degree for element labels expressed as a sentence or phrase. To this end, we use a simple measure from the natural language processing. The method takes two phrases or short sentence as input and computes a semantic similarity as output in four macro steps:

Step 1: Tokenize the names of both labels. We token each label as a set of words. Denote by *s.tok* as the *j*th token in the name label*s*.
Step 2: Search the synset of each token applying WordNet. Denote by $syn(w)$ the WordNet synset of a word w.
Step 3: Compute the Semantic similarity. We use the Jaccard similarity to calculate the Semantic similarity on the synsets of each pair of tokens.

Step 4: Return the average-max Semantic similarity as the result. The formula is shown as follows:

$$sim_{se}(s,t) = \frac{1}{|s.tok|}\sum_{i}\max_{j}(Jaccard(syn(s.tok_i), syn(t.tok_j))). \quad (2)$$

Here $|s.tok|$ is the number of tokens in the name of s, *Jaccard* denotes the Jaccard similarity between two sets, and $syn(w)$ denotes the WordNet synset of a token w.

4.3 Structural Similarity Function

Inspired by the work in [23], we propose a structural similarity function for matching RDF graph, in which we further take edge information into consideration.

4.3.1 Structural similarity function

Let v_s is a vertex of fuzzy RDF graph G_s, *i.e.* $v_s \in G_s$, $v_s' \in pre(v_s)$, and $e_s \in E_s$ is a directed edge from vertex v_s to vertex v_s'. Our matching method iteratively computers a similarity degree for every vertex pair (v_s, v_t) of two fuzzy RDF graphs by aggregating the similarity degrees between the immediate neighbors of v_s and v_t. By neighbors, we mean either successor or predecessor, depending on which bi-similarity notion is being used. The method iterates until either the similarity degree between all vertex pairs stabilize or a maximum number of iterations is reached.

If the similarity of edge e_s is different from the similarity of e_t or the fuzzy membership ρ_s deviates far from the fuzzy membership ρ_t, the similarity degree of v_s' and v_t' will have less effect on the similarity degree of v_s and v_t. For proper aggregation of similarity degrees between vertices of two fuzzy RDF graphs, we further take edge labels similarity and edge fuzzy membership into consideration. We compare edge labels using label similarity functions defined in section 4.2. These similarity functions assign a similarity degree $sim_e(e_s, e_t)$ to every edge labels pair (e_s, e_t). We use a measure of similarity of fuzzy membership degrees [5], which is based on the difference as well as the sum of corresponding grades of membership. Let ρ_s and ρ_t are the fuzzy membership degrees of edges e_s and e_t, respectively. The similarity of fuzzy values ρ_s and ρ_t is defined by

$$sim_\rho(\rho_s, \rho_t) = 1 - \frac{(\rho_s \vee \rho_t) - (\rho_s \wedge \rho_t)}{\rho_s + \rho_t} \quad (3)$$

Here $(\rho_s \vee \rho_t) = \max(\rho_s, \rho_t)$ and $(\rho_s \wedge \rho_t) = \min(\rho_s, \rho_t)$.

We now describe the computation of forward similarity. For every vertex pair (v_s, v_t), the similarity degree $Sim^i(v_s, v_t)$ is computed from (1) similarity degree between v_s and v_t after step $i-1$, *i.e.*,$Sim^{i-1}(v_s, v_t)$; (2) similarity degree between the forward neighbors of v_s and those of v_t after step $i-1$, *i.e.*,$Sim^{i-1}(v_s', v_t')$; (3) similarity degree between the edge labels relating v_s and v_t to their forward neighbors, *i.e.*, $sim_e(e_s, e_t)$; (4) similarity degree between the edges fuzzy values of e_s and e_t.

To find the best match for v_s among the forward neighbors of v_t, we need to maximize the value $sim_e(e_s, e_t) \times sim_\rho(\rho_s, \rho_t) \times Sim^{i-1}(v_s', v_t')$. The similarity degrees between the forward neighbors of v_s and their best matches among the forward neighbors of v_t

after the *i*th iteration are computed by

$$sim^i(v_s, v_t) = \frac{1}{|pre(v_s)|} \sum_{v_s' \in pre(v_s)} \max_{v_t' \in pre(v_t)} sim_e(e_s, e_t) \times sim_\rho(\rho_s, \rho_t) \times Sim^{i-1}(v_s', v_t'). \quad (4)$$

And the similarity degrees between the forward neighbors of v_t and their best matches among the forward neighbors of v_s after iteration i are computed by

$$sim^i(v_t, v_s) = \frac{1}{|pre(v_t)|} \sum_{v_t' \in pre(v_t)} \max_{v_s' \in pre(v_s)} sim_e(e_t, e_s) \times sim_\rho(\rho_t, \rho_s) \times Sim^{i-1}(v_t', v_s') \quad (5)$$

Note that this *sim* measure is asymmetric, *i.e.*, $sim^i(v_s, v_t) \neq sim^i(v_t, v_s)$.

In conclusion, we define the forward similarity degrees of vertex pair (v_s, v_t) after the *ith* iteration as follows:

$$Sim^i(v_s, v_t) = ((sim^i(v_s, v_t) + sim^i(v_t, v_s))/2 + Sim^{i-1}(v_s, v_t))/2. \quad (6)$$

For backward similarity degrees calculating, we perform the above formula for vertex v_s and v_t, but consider their backward neighbors instead of their forward neighbors.

4.3.2 Iterative computation

To calculate $sim(v_t, v_s)$ from forward neighbors, we present an iteration method by iteratively applying Formula (6). The computation has two phases: the initialization phase which assigns $sim^0(v_t, v_s)$ for every vertex pair (v_t, v_s), and the iteration phase which update the degree of $sim^i(v_t, v_s)$ by using $sim^{i-1}(v_t, v_s)$ according to Formula (6), when $i \geq 1$. The principle of the method is that the similarities between two vertices must depend on the similarities between their adjacent vertices. We summarize the procedure of iterative computation of forward similarity in Algorithm 1 as following.

Algorithm 1: Structural similarity algorithm

Input: two fuzzy RDF graphs G_S and G_T, and a constant ε
Output: matching similarity *Sim*

1: for each $v_s \in V_s$, $v_t \in V_t$ do
2: $\rho(\hat{v}_s, v_s) \leftarrow \mu(v_s)$ and $\rho(\hat{v}_t, v_t) \leftarrow \mu(v_t)$
3: for each $e_s \in V_s \times \{\hat{v}_s\}$, $e_t \in V_t \times \{\hat{v}_t\}$ do
4: $sim(e_s, e_t) \leftarrow 1$
5: for each $v_s \in V_s$ and $v_t \in V_t$ do
6: if $(v_s = \hat{v}_s)$ and $(v_t = \hat{v}_t)$ then
7: $Sim^0(v_s, v_t) \leftarrow 1$
8: else
9: $Sim^0(v_s, v_t) \leftarrow 0$
10: repeat
11: $i \leftarrow i + 1$
12: for each $v_s \in V_s$ and $v_t \in V_t$ do
13: $Sim^i(v_s, v_t) \leftarrow ((sim^i(v_s, v_t) + sim^i(v_t, v_s)) / 2 + Sim^{i-1}(v_s, v_t)) / 2$
14: until $|Sim^i(v_s, v_t) - Sim^{i-1}(v_s, v_t)| < \varepsilon$
15: return *Sim*

Algorithm 1 is an iteration method by iteratively applying Formula (6). We begin by assigning the fuzzy degree $\mu(v)$ associated with each vertex $v \in V$ except $\hat{v}$ to edges (v, $\hat{v}$) and ($\hat{v}$, v) in lines 1-2. Then we initialize the similarity $sim^0(\hat{e}_s, \hat{e}_t)$ for the edge pair $(\hat{e}_s, \hat{e}_t)$ formed by the home vertices $\hat{v}_s$ or $\hat{v}_t$ connected with other real vertices in lines 3-4. The similarity $sim^0(\hat{e}_s, \hat{e}_t)$ is set to 1.0 since their edges labels are inexistence. At the same time, we initialize the similarity for vertex pair in lines 5-9. For every home vertex pair $(\hat{v}_s, \hat{v}_t)$, we set $sim^0(\hat{v}_s, \hat{v}_t) = 1.0$ in line 7. For every other vertex pair (v_s, v_t), we assign the similarity degrees $sim^0(v_s, v_t) = 0$ as initial similarities in line 9. We further use iteration method to calculate the matching similarity in lines 10-14. In each iteration, we update the similarity degree for each vertex pair (v_s, v_t) using the similarities of their neighbors in the previous iteration. The value $Sim^i(v_s, v_t)$ is non-decrease as i increase, and the method iterates until either the similarity degree between all vertex pairs stabilize or a maximum number of iterations is reached. Note that, the similarities between home vertices and real vertices are not refreshed during the iteration. Finally, we return the matching similarity Sim as the result in line 15.

4.3.3 Convergence

Lemma 1: The iterative formula (6) is bounded in the interval [0, 1].

***Proof*:** This follows the analysis and the definitions of the iterative formula. It is quite straightforward. For all $v_s \in V_s$, $v_t \in V_t$, $i \geq 1$, $Sim^i(v_s, v_t) \in [0, 1]$, *i.e.*, iterative formula is bounded.

Lemma 2: The iterative formula (6) is monotone non-decreasing.

***Proof*:** This can be simply proved by mathematical induction.

Basis: Let show that the monotone holds for $i = 1$. If $\hat{v}_s \in pre(v_s)$ and $\hat{v}_t \in pre(v_t)$, we have $sim^1(v_s, v_t) \in [0, 1]$, otherwise, $sim^1(v_s, v_t) = 0$.

Similarly, we have $sim^1(v_t, v_s) \in [0, 1]$. According to the iterative formula (6), $Sim^1(v_s, v_t) \in [0, 1]$.

Since the initial similarities $sim^0(v_s, v_t) = 0$, we have $Sim^0(v_s, v_t) = 0$. Thus $Sim^0(v_s, v_t) \leq Sim^1(v_s, v_t)$. Therefore, the monotone non-decreasing holds for $i = 1$.

Inductive Step: Assume $Sim^{k-1}(v_s, v_t) \leq Sim^k(v_s, v_t)$ holds for $i = k$. According to the above iterative formula definitions, we have

$$sim^k(v_s, v_t) = \frac{1}{|pre(v_s)|} \sum_{v_s' \in pre(v_s)} \max_{v_t' \in pre(v_t)} sim_e(e_s, e_t) \times sim_\rho(\rho_s, \rho_t) \times Sim^{k-1}(v_s', v_t')$$

$$\leq \frac{1}{|pre(v_s)|} \sum_{v_s' \in pre(v_s)} \max_{v_t' \in pre(v_t)} sim_e(e_s, e_t) \times sim_\rho(\rho_s, \rho_t) \times Sim^k(v_s', v_t') = sim^{k+1}(v_s, v_t).$$

Thus, we have $Sim^k(v_s, v_t) \leq Sim^{k+1}(v_s, v_t)$, that is, the monotone non-decreasing holds for $i = k + 1$.

Since both the basis and the inductive step have been performed, by mathematical induction, the monotone non-decreasing holds for all $i \geq 1$.

Theorem 1: Iterative formula (6) is convergence.

***Proof*:** Based on the above Lemma 1 and Lemma 2, it is obviously proved.

4.4 Combining Similarities and Alignment Extraction

In order to obtain the overall similarity degrees between vertices, we need to aggregate the results of different similarity functions. There are several approaches to this, including linear averages, nonlinear averages and machine learning techniques. In our works, we use a simple approach based on linear averages. Firstly, we obtain the label similarity (Sim_L) by taking an average of the syntactic similarities (sim_{sy}) and semantic (sim_{se}) similarities. Then, total similarity (Sim) is calculated by combining the label similarity and structural similarity (Sim_S). To more accurately distinguish between similarity scores that are close to the median, we use a non-linear function, sigmoid function [19], to compute each similarity score. The idea behind using a sigmoid function is quite simple: it allows reinforcing similarity scores higher than 0.5 and to weaken those lower than 0.5. That is to say, the sigmoid function provides high values for the best matches and lower ones for the worse matches. This treatment is meant to clearly separate two zones: the positive and negative correspondences. In this way, the general formula for this combination task can be given as following:

$$Sim(v_s, v_t) = \omega\, sig(Sim_L(v_s, v_t)) + (1-\omega)\, sig(Sim_S(v_s, v_t)). \quad (7)$$

Here ω is a pre-defined weight, $sig(x) = \frac{1}{1+e^{-\alpha(x-0.5)}}$, α being a parameter for the slope.

To obtain a correspondence relation between input fuzzy RDF graphs, we set a threshold δ for translating the overall similarity degrees into a binary relation. Pairs of data with similarity degree above the threshold are included in and the rest are left out. However, the choice of the thresholds δ is difficult: an increment of δ results in increased matching quality (*i.e.*, a low number of false positives), but simultaneously reduces the matching coverage (*i.e.*, a low number of false negatives). Similarly, a smaller δ decreases the matching quality along with a higher matching coverage (see the experiments in Section 5.3 for the effect of varying δ). In practice, we expect to produce a small decrease the matching quality if it can bring about a comparable increase the matching coverage. Because it is easier for us to remove incorrect matches rather than find the missing ones in the process of data matching.

5. EXPERIMENTAL EVALUATIONS

5.1 Data Sets

We generate the synthetic dataset based on the Person-Restaurant collection proposed in the instance-matching track of the Ontology Alignment Evaluation Initiative [13]. The collection contains 3 pairs of datasets. Table 1 provides importing information

for any dataset: number of vertices ($|V_s|$ column), number of generated edges ($|E_s|$ column), and number of correct matches. Moreover, to describe fuzzy information, we make use of a random data generation method that transforms the ordinary RDF into a fuzzy data set.

We implement all the algorithms in Java and run on Window 7 system with 2.5 GHz Intel(R) Core(TM) i5 processor and 8 GB RAM.

Table 1. Importing information for any datasets evaluated.

Datasets	Source Graph		Target Graph		Number of reference alignments
	$\|V_s\|$	$\|E_s\|$	$\|V_t\|$	$\|E_t\|$	
Person11-Person12	58	121	43	104	38
Restaurant1-Restaurant2	34	75	29	56	25
Person21-Person22	48	77	39	68	31

5.2 Evaluation Metrics

A matching approach is effective if it neither produces too many false positives nor misses too many false negatives. This paper uses the classical criterion [15]: *Precision*, *Recall* and *f-measure* to evaluate the answer quality. Let $|tp|$ be the number of correct matches produced by a matching approach, $|fp|$ be the number of matches found by matching approach that does not exist in the correct matches, and $|fn|$ be the number of matches not found by matching approach. We can define $Precision = |tp| / (|tp| + |fp|)$, $Recall = |tp| / (|tp| + |fn|)$ and $f\text{-}measure = 2 \times Precision \times Recall / (Precision + Recall)$.

5.3 Evaluation Results

To analyze the effectiveness of our proposed approach, we perform three groups of experiments. We first investigate the performances of the combination matching (abbr. CM) which combine these results of label similarity matching (abbr. LM) and structural similarity matching (abbr. SM). And we obtain a desired threshold δ_m, for a good trade-off between effectiveness, by using different thresholds ranging from 1 down to 0.5. Then we compare the performance of these three methods in different data sets. Besides the match quality, we also evaluate time cost of the matching algorithm.

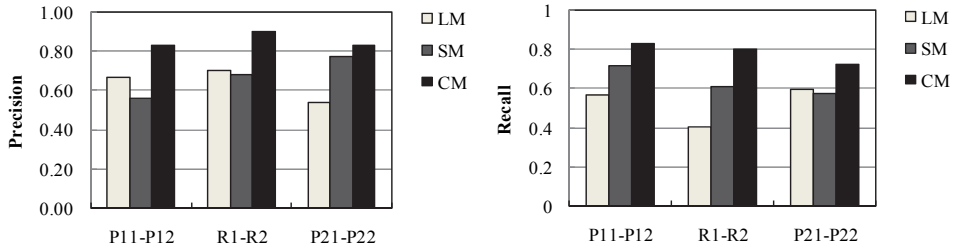
In the first experiment, we study how affect the overall performance of CM by using different thresholds δ. As introduced in Section 4.4, threshold refers to the cutoff value used for determining the correspondence relation from the similarity degrees. A reasonable threshold is vital in the process of matching. To determine an appropriate value for δ, we calculate CM's *Precision* (P), *Recall* (R) and *f-measure* (f) for thresholds ranging from 0.5 to 1, in increments of 0.05. The results are presented in Table 2.

As we can observe, the standard deviation (SD) of the *Precision* and *Recall* are close to 0.1, meaning that the different data sets do not affect much the CM's overall performance. Moreover, we use the median, which is a commonly used measure for the properties of a data set in statistics, to define an appropriate value for parameters δ. As we can see, the overall performance of CM is relatively well when threshold $\delta_m = 0.75$. Therefore, for all other experiments shown in this paper, we will use in our evaluation: $\delta_m = 0.75$.

Table 2. CM's performance varying δ.

Data	Person11-Person12			Restaurant1-Restaurant2			Person21-Person22		
δ	P	R	f	P	R	f	P	R	f
1	0.706	0.756	0.730	0.812	0.857	0.834	0.701	0.681	0.691
0.95	0.743	0.386	0.508	0.804	0.903	0.851	0.620	0.772	0.688
0.90	0.710	0.552	0.621	0.791	0.770	0.780	0.518	0.810	0.632
0.85	0.778	0.687	0.730	0.883	0.737	0.803	0.563	0.745	0.641
0.80	0.659	0.720	0.688	0.890	0.874	0.882	0.721	0.709	0.715
0.75	0.833	0.833	0.833	0.902	0.802	0.849	0.827	0.727	0.774
0.70	0.704	0.683	0.693	0.813	0.64	0.716	0.645	0.623	0.634
0.65	0.663	0.781	0.717	0.882	0.717	0.791	0.519	0.751	0.614
0.60	0.510	0.880	0.646	0.756	0.630	0.687	0.470	0.554	0.509
0.55	0.535	0.817	0.647	0.709	0.403	0.514	0.540	0.670	0.598
0.50	0.527	0.882	0.660	0.620	0.802	0.699	0.407	0.716	0.519
SD	0.106	0.149	0.081	0.086	0.142	0.105	0.123	0.072	0.079

In the second group experiments, we compare the match quality of LM, SM and CM over the above three data sets. Fig. 3 (a) represents the *Precision* values of the three methods, and Fig. 3 (b) represents the *Recall* values for those case studies. Bars refer each method, using different gray scales, *i.e.*, from LM, white bars, to CM, black bars. For the convenience of diagram representations, we abbreviate the data sets name Person11-Person12, Restaurant1-Reataurant2 and Person21-Person22 to P11-P12, R1-R2 and P21-P22, respectively.



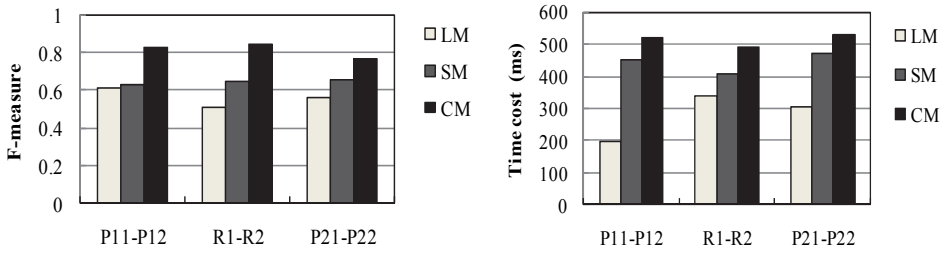
(a) Precision (b) Recall

Fig. 3. Results of LM, SM, and CM matching methods.

It can be seen that the precision of LM is higher than SM in the P11-P12 and R1-R2 data pairs, where elements with typographically similar names are likely to correspond. However, LM misses several correct matches, and hence has low recall. SM, in contrast, has lower precision, but high recall. Furthermore, the performance of LM is poor than SM in the P21-P22 pairs. This observation can be explained by the fact that these datasets were built by adding spelling mistakes or null value to the elements and literals values of their original datasets. While SM is able to exploit better the similarity in misspelled data or blank vertices by structural information. Moreover, we can see that the

precision rates of our proposed CM are higher than all the existing evaluated approaches to all the data sets. For example, the precision values for the LM, SM, and CM in the R1-R2 are 0.703, 0.682, and 0.902, respectively. This is explained by the fact that LM and CM approaches are filtering out each other's false positives. Recall remains high in the CM approach, as LM and SM find many complimentary high-quality matches.

Finally, in the third group of experiments, we compare the execution time of CM against LM and SM. Results for these approaches are reported in Fig. 4. As observed in Fig. 4 (a), we find that a structural similarity metric based on graph matching achieved the highest retrieval quality (*f-measure*). However, the metric is hindered by the fact that the iterative computation is time-consuming. Fig. 4 (b) represents the time costs of LM, SM, and CM in different data sets. As we can see, corresponding time cost of SM is no more than the double of LM's and lowers than that of CM. It is not surprising owing to the high complexity of computing graph structural similarities. Therefore, we need improvement in terms of time by decreasing the number of iterations that strike a tradeoff between computational complexity and precision of matching in practice.



(a) f-measure (b) Execution time

Fig. 4. Time cost of LM, SM, and CM matching methods.

6. CONCLUSIONS

RDF data matching in the context of fusion and interlink of RDF datasets is the key task of determining if two resources are referred to the same entity in the real world. In this paper, we propose an efficient fuzzy RDF graph matching approach based on the similarity measure, which takes textual characteristics and structural features into account. Firstly, label matching is calculated by combining syntactic and semantic similarities of labels of vertices or edges. We then propose a function computing structural similarity, in which we further take edge similarity and edge fuzzy value into consideration. Next, we combine these results of the individual metrics to obtain the total similarities. Finally, we extract the correspondences based on the total similarities.

While the evaluation results demonstrate the effectiveness of our approach, it has a limitation. As shown in Section 4.5, the time complexity of computing structural similarity depends on several factors such as the size and the density of the RDF graph, and the numbers of iterations. We intend to investigate the evaluation of convergence so that the iteration number in calculating similarity can be reduced, which is a matter of obvious importance when handling large RDF graphs. Moreover, it is worth of future research to apply Granular Computing techniques to solve the problem of fuzzy RDF data matching.

REFERENCES

1. M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida," *Journal of the American Statistical Society*, Vol. 84, 1989, pp. 414-420.
2. A. Nikolov, V. Uren, and E. Motta, "Towards data fusion in a multi-ontology environment," in *Proceedings of Workshop on Linked Data on the Web*, 2009. pp. 20-24.
3. A. Skowron, A. Jankowski and S. Dutta, "Interactive granular computing," *Granular Computing*, Vol. 1, 2016, pp. 95-113.
4. C. F. Dorneles, R. Gonçalves, and R. D. S. Mello, "Approximate data instance matching: a survey," *Knowledge and Information Systems*, Vol. 27, 2011, pp. 1-21.
5. C. P. Pappis and N. I. Karacapilidis, "A comparative assessment of measures of similarity of fuzzy values," *Fuzzy Sets and Systems*, Vol. 56, 1993, pp. 171-174.
6. D. Zhang, T. Song, J. He, X. Shi, and Y. Dong, "A similarity-oriented RDF graph matching algorithm for ranking linked data," in *Proceedings of International Conference on Computer and Information Technology*, 2012, pp. 427-434.
7. F. Manola and E. Miller, "RDF primer," W3C working group recommendation, http://www.w3.org/TR/rdf-primer/, 2004.
8. G. Klyne and J. J. Carroll, "Resource description framework (RDF): Concepts and abstract syntax," W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.
9. G. Peters and R. Weber, "DCC: A framework for dynamic granular clustering," *Granular Computing*, Vol. 1, 2016, pp. 1-11.
10. G. Wilke and E. Portmann, "Granular computing as a basis of human-data interaction: A cognitive cities use case," *Granular Computing*, Vol. 1, 2016, pp. 181-197.
11. H. Huang and C. Liu, "Query evaluation on probabilistic RDF databases," in *Proceedings of International Conference on Web Information Systems Engineering*, 2009, pp. 307-320.
12. H. Zhu, J. Zhong, J. Li, and Y. Yu, "An approach for semantic search by matching RDF graphs," in *Proceedings of International Florida Artificial Intelligence Research Society Conference*, 2002, pp. 450-454.
13. J. Euzenat, A. Ferrara, W. R. V. Hague, L. Hollink, C. Meilicke, and A. Nikolov, "Results of the ontology alignment evaluation initiative 2010," *Ontology Alignment Evaluation Initiative*, Vol. 814, 2012, pp. 73-95.
14. J. J. Carroll, "Matching RDF graphs," in *Proceedings of International Conference on the Semantic Web*, 2002, pp. 5-15.
15. J. M. McGill and G. Salton, *Introduction to Modern Information Retrieval*, McGraw-Hill, NY, 1983.
16. S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: A versatile graph matching algorithm and its application to schema matching," in *Proceedings of International Conference on Data Engineering*, 2002, pp. 117-128.
17. J. N. Mordeson and P. S. Nair, *Fuzzy Graphs and Fuzzy Hypergraphs*, Physica-Verlag Heidelberg, NY, 2008.
18. J. R. Ullmann, "An algorithm for subgraph isomorphism," *Journal of the Assocmtl on for Computing Machinery*, Vol. 23, 1976, pp. 31-42.

19. M. Ehrig and Y. Sure, "Ontology mapping-an integrated approach," in C. J. Bussler, J. Davies, D. Fensel, and R. Studer, ed., *The Semantic Web: Research and Applications*, Springer-Verlag Berlin Heidelberg, 2004, pp. 76-91.
20. M. Mazzieri, A. F. Dragoni, and U. P. D. Marche, "A fuzzy semantics for semantic web language," in *Proceedings of International Conference on the Semantic Web*, 2005, pp. 12-22.
21. O. Pivert, O. Slama, and V. Thion, "An extension of SPARQL with fuzzy navigational capabilities for querying fuzzy RDF data," in *Proceedings of IEEE International Conference on Fuzzy Systems*, 2016, pp. 2409-2416.
22. S. Araujo, J. Hidders, D. Schwabe, and A. P. D. Vries, "Serimi-resource description similarity, RDF instance matching and interlinking," in *Proceedings of International Conference on Semantic Web*, 2011, pp. 246-247.
23. S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave, "Matching and merging of variant feature specifications," *IEEE Transactions on Software Engineering*, Vol. 38, 2012, pp. 1355-1375.
24. T. Pedersen, S. Patwardhan, and J. Michelizzi, "WordNet:: Similarity-measuring the relatedness of concepts," in *Proceedings of Conference on Artificial Intelligence*, 2004, pp. 1024-1025.
25. U. Straccia, "A minimal deductive system for general fuzzy RDF," in *Proceedings of International Conference Web Reasoning and Rule Systems*, 2009, pp. 166-181.
26. V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, Vol. 10, 1966, pp. 707-710.
27. W. E. Moustafa, A. Kimmig, A. Deshpande, and L. Getoor, "Subgraph pattern matching over uncertain graphs with identity linkage uncertainty," in *Proceedings of IEEE International Conference on Data Engineering*, 2014, pp. 904-915.
28. X. Lian and L. Chen, "Efficient query answering in probabilistic RDF graphs," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2011, pp. 157-168.
29. X. Zhu, S. Song, X. Lian, J. Wang, and L. Zou, "Matching heterogeneous event data," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2014, pp. 1211-1222.
30. Y. Fukushige, "Representing probabilistic relations in RDF," in *Proceedings of International Semantic Web Conference*, 2005, pp. 106-107.
31. Y. Liu, F. Scharffe, and C. Zhou, "Towards practical RDF datasets fusion," in *Proceedings of Asian Semantic Web Conference Workshop on Data Integration through Semantic Technology*, 2008, pp. 77-86.
32. Y. Lv, Z. M. Ma, and L. Yan, "Fuzzy RDF: A data model to represent fuzzy metadata," in *Proceedings of International Conference on Fuzzy Systems*, 2008, pp. 1439-1445.
33. Y. Raimond, C. Sutton, and M. Sandler, "Automatic interlinking of music datasets on the semantic web," in *Proceedings of the WWW2008 Workshop on Linked Data on the Web*, 2008, pp. 137-145.
34. Z. M. Ma, J. Liu, and L. Yan, "Matching twigs in fuzzy xml," *Information Sciences*, Vol. 181, 2011, pp. 184-200.

Guan-Feng Li received the M.S. degree in Computer Software and Theory from Ningxia University, China. He is currently pursuing the Ph.D. degree in the School of Computer Science and Engineering, Northeastern University, China. His research interests include RDF data and semantic web.

Zong-Min Ma is currently a Full Professor at Nanjing University of Aeronautics and Astronautics, China. He received his Ph.D. degree from the City University of Hong Kong, China. His research interests include databases, the semantic web, and knowledge representation and reasoning with a special focus on information uncertainty. He has published more than one hundred and seventy papers on these topics. He is also the author of four monographs published by Springer. He is a senior member of the IEEE.