Bipartite graph model for RDF data cleansing

Huang Li[1, 2*]

[1] *College of Computer Science and Technology, Wuhan University of Science and Technology*

[2] *Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System*

Received 1 March 2014, www.tsi.lv

Abstract

Many systems use RDF to describe information resources and semantic associations between resources. RDF data plays a very important role in advanced information retrieval. Due to diversity and imprecise of resources, duplicates exist in RDF data. The query and retrieval of RDF data are studied by many researchers. However, researchers seldom study RDF data cleansing. In this paper, we focus on RDF data cleansing. According to the features of RDF data, we propose a new approach. This approach combines similarity and connections among resources. First, we introduce an intermediate model, named RDF-Bipartite Graph model, to represent the RDF data. This model improves from Bipartite Statement-Value Graphs model. Then on the proposed model, we design a Subgraph-Extend method, to find the path connecting two nodes. This method detects the minimum subgraph containing two nodes for connect-path finding. It avoids the connect-weight setting in traditional method. Experiments on publication datasets show that the proposed method is efficient in duplicate detection of RDF data, and has high performance and accuracy.

Keywords: RDF, data cleansing, Bipartite Graph

1 Introduction

Revolution of information technology has led to an explosion in the growth of data in digital form. Then this incurs a problem: how to manage and use such large amount of data. One difficulty to manage those data is theirs diverse sources, for example, the data are managed by different organization and described and stored in different form, although data integration systems and search engines can help users manage them in some sense. However, due to diversity of data resources, there is a large amount of duplicates after data integration.

The semantic associations among resources are as the same important for information querying and retrieval as information resources themselves. Many systems use RDF (Resource Description Framework) data to represent resources and the semantic associations among resources. Due to the existence of duplicates in different resources, duplicate still exist in RDF data integration. Thus RDF data need to be cleaned.

We propose an approach to detect duplicates in RDF data. The duplicate detection method in this approach combines the relationships and similarities among resources. To extract more information of relationships among resources, we improve the Bipartite Statement-Value Graph model for RDF [1]. We call the new intermediate model RDF-Bipartite Graph Model. The relationships are presented as connecting path among data in this model. We also propose a method, Subgraph-Extend method, to find a path between two nodes in RDF-Bipartite Graph. This method utilizes the attributes of RDF-Bipartite Graph. This method extends subgraph to find the connect path, instead of find shortest path using the sum of weights of edges. Then the resistance of connect-path is used to detect duplicates instead of connect-weight.

The rest of the paper is organized as follows. The RDF-Bipartite Graph model is defined in section 2. Section 3 describes the method of finding connect path among data. In section 4, we report the performance of our approach. Related work is discussed in Section 5. Finally, we conclude this paper in section 6.

Your goal is to simulate the usual appearance of papers in a Journal of the Academy Publisher. We are requesting that you follow these guidelines as closely as possible.

2 Bipartite Graph Model for RDF Data

2.1 RDF DATA

RDF (Resource Description Framework) is proposed by WWW Consortium. It is a language to describe metadata about information resource. RDF statement is a triple. It is consisted of a subject, a predicate and an object. A triple means to assign a value (subject) to one kind property (predicate) of the resource (object). The subject is the resource, which is described, the predicate describes some property, and the object is the value of the property. A set of RDF statements composes a RDF Graph. Figure 1 shows part of RDF graph.

* Corresponding author - Tel: +86-130-988-82821; fax: +86-027-6889-3420;
E-mail: huangli82@wust.edu.cn

In Figure 1, we can find out that an RDF graph is not a simple direct graph. There may be more than one edges between two nodes. If we only use a direct graph to represent the RDF data, some information must be missing. In RDF graph, each edge refers to an RDF statement. Every edge connects three nodes. Therefore, the RDF graph is a hypergraph. In a hypergraph, the hyperedges express the statements, and the hypernodes denote subjects, predicates and objects.

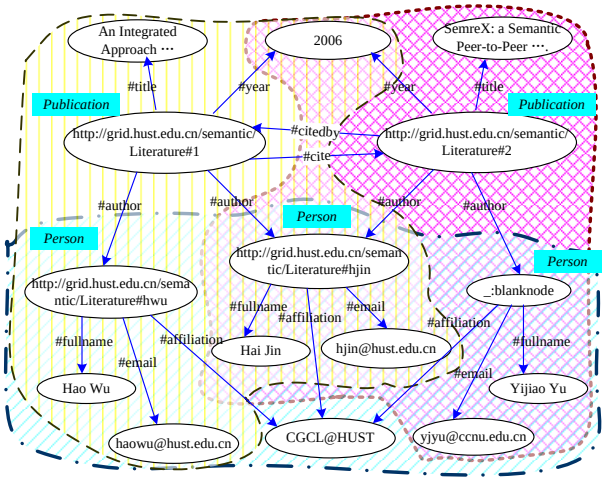

FIGURE 1 Part of RDF graph

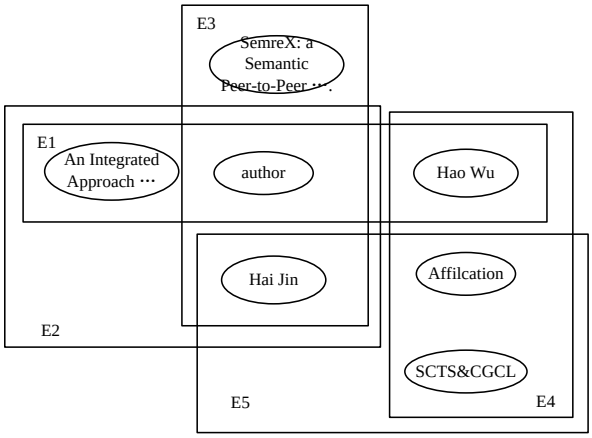

FIGURE 2 Part of hypergraph of Figure 1

Figure 2 shows a part of hypergraph of figure 1. There are seven hypernodes denoted by elliptoid point and five hyperedges: E1, E2, E3, E4 and E5. All hyperedges connect three nodes.

2.2 RDF-BIPARTITE GRAPH MODEL

In [1], Hayes et al. proposed bipartite graphs model to represent RDF data. In the bipartite graphs model, both statement and value are represented by a node in the graph. They transform the RDF graph into a bipartite graph. The two sides of nodes are statement nodes and value nodes. The bipartite graph is a hierarchical graph. Moreover, the entities in the same layer have strong relationship. All subjects, objects and predicates are called as value statements.

The hierarchy bipartite graph is structured as follows. Because the subject and object of one statement are in one side of bipartite graph, the subject and object are in the same layer, in each statement. The predicate is in the other layer. Moreover, we rule the layer of predicate is higher.

The path between two nodes of same layer in RDF-Bipartite graph is referred the degree of connection between them. If the path is shorter and weightier, the connection is stronger.

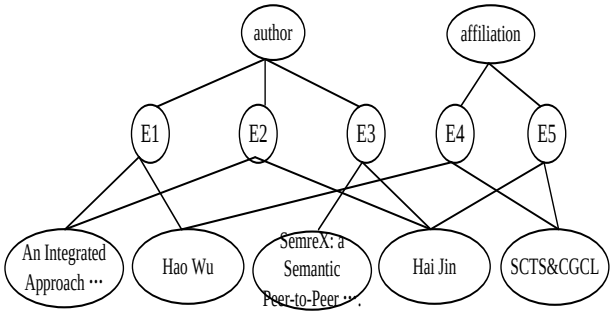

FIGURE 3 Bipartite graph of naïve hypergraph in Figure 2

For duplicate detecting, the relationships among duplicates are indirect. Duplicates must be of the same type. The subjects and the objects are of the same type. Therefore, the duplicates are on the same layer. Thus, the path is found in the same layer. That is the nodes in the paths must be in the same and higher layers. Duplicates may exist in value nodes. The statement nodes are the bridges of value nodes. We name the above path, d-path, for distinguishing with other paths in the graph.

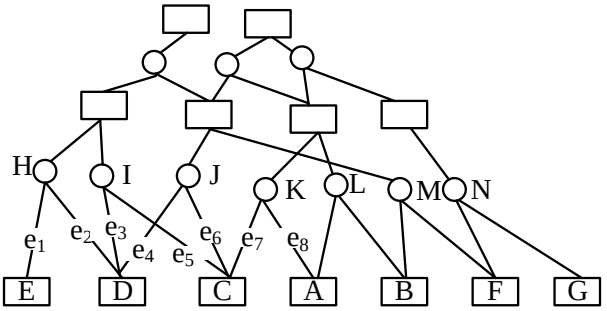

FIGURE 4 Toy RDF-Bipartite graph. The circular points denote statement nodes. And value nodes are denoted by rectangle points.

For example, in figure 4, the paths from ‘E’ to ‘A’ are $\{(e_1,e_2,e_3,e_5,e_7,e_8), (e_1,e_2,e_4,e_6,e_7,e_8)\}$. The d-paths connecting two value nodes in the same layer are not unique.

First, we give some definitions.

Definition 1. Connect-weight. **C**onnect-weight is to measure the connective level of path, which connect two nodes. It is the sum of all the edges weights in the path.

Definition 2. Connect-Path. For all paths connecting two nodes, the path which has the biggest connect-weight is called connect-path.

Then the problem changes to find the connect-path between two nodes.

In an RDF-Bipartite graph model, because all RDF statements are consisted of 3-tuple, the degree of statement nodes are three. Moreover, according to the characteristic of RDF-Bipartite graph model, only subject and object of a statement are in the same layer. If a path passes through statement node, it must be preceded by subject and followed by object of the statement, or preceded by object and followed by subject of the statement.

In next section, we will propose a method to find connect-path between two nodes in RDF-Bipartite graph.

3 Connect-Path

3.1 SUBGRAPH-EXTEND

Before describing SE, we will introduce some new concepts used in SE. The subgraph in this method is a special subgraph. This subgraph is consisted of units instead of nodes. There can be many nodes in a unit.

According to the definition of RDF-Bipartite, RDF-Bipartite graph is consisted of 3-tuples. We can take a 3-tuple as a unit in RDF-Bipartite. Figure 5(a) shows a unit in a subgraph of RDF-Biparitite. And Figure 5(b)(c) shows different kinds of subgraphs in RDF-Bipartite.

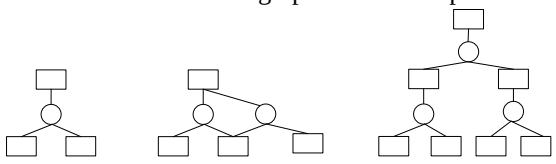

FIGURE 5 Subgraphs of RDF-Bipartite

Fig. 5(b) is consisted of two units, and (c) has three units. In SE, one subgraph is extended by units. Fig 5(b) can be considered as extended from 5(a), and 5(c) could be extended from 5(a) or 5(b).

In fig 5, we can also find that the units in one subgraph overlap with each other. If the units are independent, the subgraph which are consist by these units is not a connective subgraph. This subgraph is to help finding connect-path. And the subgraphs are all connect graph. Thus, in SE, we extend the subgraph by units, which overlap with the exiting units. Obviously, the more two units overlap, the stronger connection the units have. This is the principle of SE. According to the principle, we give the extending rule in this method as follow.

***Extending Rule**:* A graph G, and G' is a subgraph of G. For any unit u, and $u \subseteq G\text{-}G'$. If $u \cap G' \neq \Phi$, u is the candidate unit for extending.

The process of the extending is iterative. The extending rule chooses the candidate unit for extending. In general, the candidate units are not unique. However, if all the candidate units are extended to the subgraph, the scale of the subgraph grows quickly. And the subgraph is closed to the graph after several iterations. Thus, we needs to choose the unit for extending from the candidate set. Every time a subgraph only extends several units. The candidate units, which have more overlaps with the subgraph, are considered first. Choosing rule is described as follows.

***Choosing Rule 1**:* A set S is the candidate units set, and the subgraph is denoted as G'. Choosing the units in the set S' and $S'=\{u \mid \max\{|u \cap G'|\}, u \subseteq S\}$.

The operation '$|x|$' is also getting the number of the element in a set. The element of the set is the node. A unit is also a subgraph of the graph.

There could be more than one candidate unit with same most overlap. Therefore, we use a set to represent the choosing units. The number of units in the choosing set may be zero, one or more than one. The number shows the units which will be added in one time. If the number is zero, this is one case to stop the extending. If the result subgraph is satisfied for the need, the iterative also stops.

3.2 CONNECT-PATH DETECTION

In previous section, we described the principles and rules of SE. We also introduce the basic method of SE. There are several different implementations for different problems. In this section, we will detect connect-path between two aim-nodes in a graph. We conclude two implementations. One is a one-side extension. It starts extending from one node and stops when the other node is in subgraph. The other implementation is a two-side extension. It starts extending from both nodes and stops when the two extending subgraphs have overlap.

The former implementation chooses the units, which are strongest connection with subgraph to extend every time. This is easy to implement, but is blindly. If the other note is in the unit, which weaker connects with the subgraph, the first extending may omit that unit. This unit may be extended after many iterative times. The connect path between the two notes in the subgraph is longer. And sometimes, this method may miss that unit. The goal of latter implementation is clear. Every extension must consider the other subgraph. Moreover, both sides can extend in parallel. This reduces the running time of the extending. Here, we choose the latter extension method.

The two-side extension starts from two nodes. It extends the two subgraphs from two nodes. The initialize subgraph is the unit, which contains the aim-node. Figure 6 shows the initialize subgraphs of toy example in Figure 4.

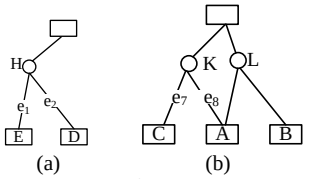

FIGURE 6 Initialize subgraphs of toy example. (a) shows the initialize subgraph of node e, and (b) shows the initialize subgraph of node a.

According to the analysis in Section 2.2, RDF-Bipartite graph is special. The connect-path must be in the same or higher layer. The candidate units must be in the same layer.

In each iterative, the extending processes from two nodes are not independent. They connect with each other. We do not only consider the connection between the candidate units and the subgraph, but also consider the subgraph extending from the other side. If the unit in both candidate sets of subgraphs extending from different sides, the unit connects both sides of subgraphs, it should be considered to be extended first. Then the subgraph contains two aim-nodes are found. The extending process could stop. According to the above analysis, we conclude another choosing rule as follows.

The graph is G, a subgraph from one side is G_1', the other side is G_2', the sets S_1, S_2 are the candidate units sets of G_1' and G_2', respectively.

Choosing Rule 2: Choosing the units in a set S' and S'= $\{u \mid \max\{|u \cap G_1'|\}, u \subseteq S_1 \cap S_2$, and $S_1 \cap S_2 \neq \Phi\}$.

Rule 2 is a supplement of rule 1. However, the two choosing rules do not have continuous relationship. When the candidate sets of two sides have some common units, the choosing rule 2 is available. And the final subgraph is $u \cup G_1' \cup G_2'$. The extending process stops. In other cases, the process extends and continue iterates following the rule 1.

After finding the subgraph, which contains two aim-nodes, the connect-path can be detected easily in the subgraph.

Let us continue to discuss the toy example in Figure 4. Figure 6 gives the initialized subgraphs for the toy example. Figure 7 shows the two candidate units sets of the two sides respectively.

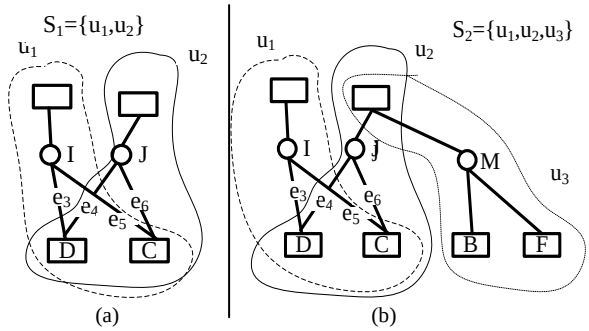

FIGURE 7 Candidate units sets. (a) is the candidate units sets of Figure6(a). (b) is the candidate units sets of Figure 6(b).

In Figure 7, we can find that unit u_1 and unit u_2 belong to both candidate units sets. Therefore, u_1 and u_2 are both in the choosing set. Because u_1 has stronger connection with Figure 6(a), the u_1 is extended. The final subgraph is shown in Figure 8.

From Figure 8, we can find the connect-path between E and A is $(e_1,e_2,e_3,e_5,e_7,e_8)$.

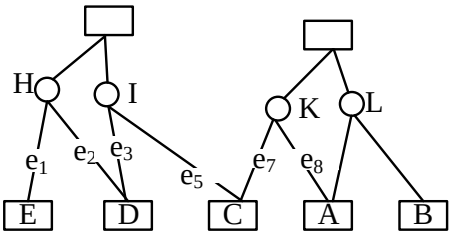

FIGURE 8 Final subgraph of the toy example

If the two nodes have weaker or no connection, the length of the connect-path between them should be larger or infinity, and the number of extending time is large or infinity. In addition, if the two nodes have weaker or no connection, they have smaller probability to be duplicates. Thus, we limit the length of the connect-path between two nodes. That is to limit the times of extending. When the number of extending times reaches the threshold value, we consider the two nodes distinct.

3.3 DUPLICATE DETECTION

However, if the two similar nodes have connect-path, they may be only similar and connective not duplicates. The probability of two similar nodes being duplicates is related to the connection between them. If two similar nodes have stronger connection, the probability that they are duplicates may be higher. Therefore, connect-weight of connect-path is also necessary in help judging. If we find the connect-path of two nodes, we can find all the units, which contain the nodes and edges of connect-path. We measure the connection between units instead of connection between nodes. The connective level between units is measured by connect-degree. The definition of connect-degree is given as follows.

Definition 3. Connect-degree. For any unit u_1, and unit u_2, the connect-degree is the number of nodes both in u_1 and u_2, denoted as $d_c(u_1,u_2)$. That is $d_c(u_1,u_2)=|u_1 \cap u_2|$.

In RDF-bipartite graph model, each unit has four elements. According to the definition of RDF-partite model, if the connect-degree is more than 2, the two nodes are the same. So the maximal connect-degree is 2. Take Fig 7(b) for example, according to the definite 3, we can get that $d_c(u_1,u_2)=2$, $d_c(u_2,u_3)=1$, and $d_c(u_1, u_3)=0$.

Because SE extends from two sides of the nodes, the more edges the connect-path has, the weaker connection the two nodes have. The connect-weight does not increase with the length of the connect-path. In addition, it may decrease while the length of the connect-path increases and the speed of decrease may increase with the length's increasing. It is hard to calculate connect-weight formally. Thus, we calculate the connect-resistance instead of connect-weight. The connect-resistance is the resistance of a path. Connect-resistance is in contrary to connect-weight. Connect-resistance is small, when the connect-weight is high. The connect-

resistance, denoted as r_c, between notes u and v is calculated by formula (1).

$$\begin{aligned} r_c(u,v) = & 1*[d_c(u,u_1)+d_c(v,v_1)]^{-1} \\ & +2*[d_c(u_1,u_2)+d_c(v_1,v_2)]^{-1} \\ & +\dots \\ & +n*[d_c(u_{n-1},u_n)+d_c(v_{n-1},v_n)]^{-1} \end{aligned}, \quad (1)$$

where $u_1,..u_n$, $v_1,...v_n$ are the nodes in the connect-path between u and v. And $u_n = v_n$. So $d_c(u_i, u_j)$ and $d_c(v_i, v_j)$ do not equal 0 simultaneously. And n is the number of extending times. It is noteworthy that the extending starts from two sides. So the length of connect-path is (2n-1).

Because $d_c(u_i, u_j)$={1, 2}, we can get

min(r_c)=(1+2+…+n)*1/[2+2]=(1+n)*n/8;

max(r_c)= (1+2+…+n)*1/[1+0]= (1+n)*n/2.

The value area of r_c is shown in Figure 9.

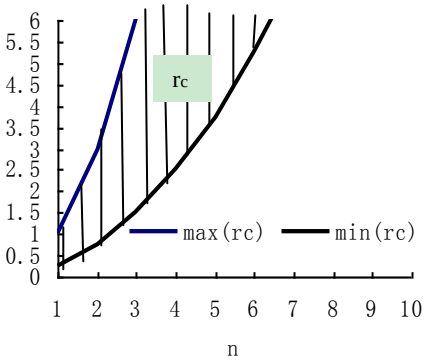

FIGURE 9 Value area of r_c

From Figure 9, the bigger n is, the longer the connect-path is, and then the connect-resistance must be bigger. And the slope of the connect-resistance's curve increases with the increasing of extending times. Along with the increase of the length of the connect-path, the connect-resistance could increase faster. So the trend of r_c conforms to reality.

If there is no path connecting two nodes, the connect-resistance is infinity. In Figure 8, the connect-path between E and A is (e_1,e_2,e_3,e_5,e_7,e_8). In Figure 10, the units u, u_1, and v contain all the nodes and edges in connect-path, and $d_c(u,u_1)$=2, $d_c(u_1,v)$=1.

Base on formula (1), the connect-resistance between E and A is $r_c(u, v)$=1*1/(2+1)=1/3.

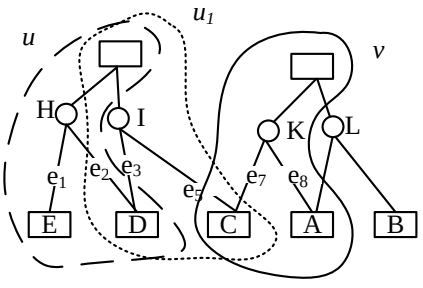

FIGURE 10 Connect units of Figure 8

If the connect-resistance is smaller than the threshold, the two nodes are considered as duplicates.

4 Experiment

In this section, we do experiments to examine the efficiency and accuracy of the proposed approach. All experiments are performed on an IBM eServer with a 1.25GHz Power4 processor and 4GB of memory, running Suse Linux Enterprise 10.0. All approaches are implemented and tested in Java.

The process of the proposed approach can be divided into three steps. The first step transfers the RDF data into RDF-Bipartite graph, like transfer fig 1. to fig 3. And then, calculate the similarities of the data pair wisely. The method of similarity calculation has been studied in [2]. All the similarity calculation pairs are in the same layer. And here, we still use the LFDW [2] to calculate similarity. If the similarity of the pair is higher than the threshold, the pair is the candidate pair. Then the candidate pairs need to be verified. The last step is to detect the connection between the data pair wisely. The connection between data is measured by the connect-path. If the connect-resistance of the connect-path between two data is smaller than a threshold, the two data are duplicates probably.

We use three popular metrics, recall, precision, and f-measure, to measure the efficiency of the approaches. In this paper, recall measures whether the approach detect all duplicates or not. Precision measures whether the detected duplicates is real duplicates or not. And F-measure is the harmonic mean of the precision and the recall rate. This metric measures the comprehensive performance.

The number of extending times is difficult to evaluate. If the number is too big, the connect-path is longer, and the connect-resistance is higher than the threshold. The two nodes are not duplicates. It wastes the running time. By conversely, if the number is too small, there are small pairs can find connect-paths. That may miss some duplicates pairs. From the analysis in Section III.C, the value of connect-resistance is in [(1+n)*n/8, (1+n)*n/2]. That means the connect-resistance is related with extending times n. The threshold thr is also related with the extending times. We can get the value of extending times satisfy with the in equation [2].

$$n \le \left\lfloor \frac{\sqrt{32*thr+1}-1}{2} \right\rfloor \quad (2)$$

If the threshold is known at the beginning, the number of extending times n is $\left\lfloor \frac{\sqrt{32*thr+1}-1}{2} \right\rfloor$. And if the threshold is unknown at the beginning, the value of the threshold is more difficult to be assigned. From Figure 9, the value of threshold is in a large scope, from

0.25 to infinity. If *n* is fixed, the threshold is fixed in the bound of [(1+*n*)**n*/8, (1+*n*)**n*/2]. Thus we assign *thr*= [(1+*n*)**n*/8+(1+*n*)**n*/2]/2 = 5*(1+*n*)**n*/16.

Then we do experiments with different extending times. The results are shown in Figure 11 and 12.

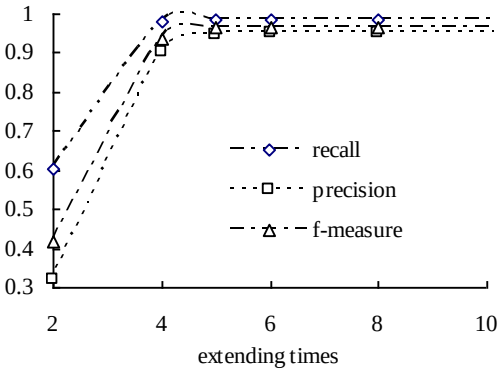

FIGURE 11 Detection results of different extending times

From Figure 11, we can find the f-measure is under 0.9 when *n* is small than 3. However, the f-measure increases quickly when *n* increases from 2 to 5. And after 5, the speed of increase is slow. The highest f-measure is get when n is 8. And when *n* is bigger than 8, the f-measure decreases a little, and keep the same value.

The f-measure is low in the beginning because *n* is too small to find the connect-path between duplicates. And along with the increase of the *n*, more connect-paths are found, so the f-measure increases quickly. Most duplicates are detected at this stage. When *n* continuously increases, there are only a few duplicates detected. Therefore, the increase speed is low. In addition, when all duplicate is detected, the f-measure is not change with *n*.

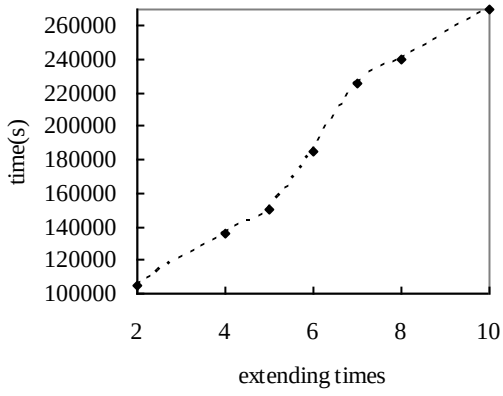

FIGURE 12 Costs of time in different extending times

Fig 12 shows the costs of time in different extending times. From figure 12, we can find the cost of time increase along the extending times increasing. This result is because more time is needed for subgraph extending to find connect-paths. And we can find the speed of increasing is not same. When n is smaller than 5, the increasing is slowly. When n is larger than 5, the increasing becomes fast. These changes are happened by the reasons as follow.

According to the analysis of Figure 11, when extending times is 5, most duplicates are detected. And along with the extending times increasing, more pairs are found having connect-paths. From Figure 11, most duplicates are found connect-path in 5 extending. When the number of extending times is lager than 5, the detecting pairs most are not duplicates, while the subgraph becomes more complex and huge; more time is needed on extending and analysing for the huge subgraph.

From Figure 11, the highest f-measure is gotten when *n* is 8. However, the f-measure increase slowly when *n* is bigger than 5. And according to Figure 12, when *n* is bigger than 5, the running time increases quickly. So in conclusion, the option value of *n* is 5 in this RDF data.

We compare the proposed approach with the just similarity method. The proposed approach combines both similarity and connection. The result is shown in figure 13. In figure 13, for simple, we use '*S*' to refer 'Similarity', and use '*C*' to refer to 'Connection'. The note '*S*+*C*' denotes the proposed approach.

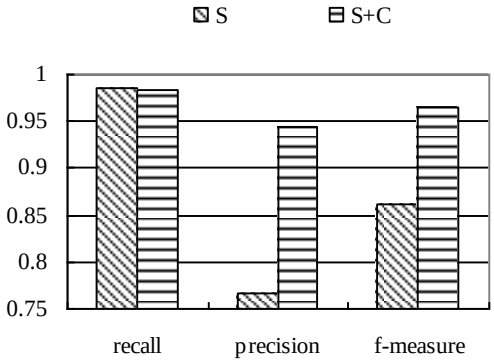

FIGURE 13 Result of comparison between S and S+C

From figure 13, we can find that, the precision of the proposed approach is higher than the method just considering similarity. This result shows that the connection among data helps for the duplicate detection. And the proposed method SE is effective for finding connection between data. In figure 13, we also find the recall of proposed method decreases tinily. This is because, the connection verifies after the filtering of similarity. So the recall of the proposed method is the same or smaller. We can assign the threshold of similarity smaller to get high recall, although more time costs for connection verifying.

5 Related work

Our research studies on solving the duplicate detecting in RDF data. This research relates with two main studies. One is duplicate detection, and the other is the model of RDF data.

Duplicate is the main inducement of data dirty. Duplicate detection is one kind of data cleansing. It is an important study in data mining. The basic method to detect duplicates is to compare entities. The main issue of compare entities is field matching [3], which could be achieved by recursive field matching algorithm [4], Smith-Waterman algorithm and R-S-W algorithm [5] etc. The above methods all consider the records themselves and omit the relationships among them. Recently, researchers shifted their attention to the associations among entities. Han et al. proposed an unsupervised learning approach using K-way spectral clustering that disambiguates authors in citations. It utilizes three kinds of citation attributes: co-author names, paper titles, and publication venue titles [6]. A general object distinction methodology is introduced in [7]. The approach combined two complementary measures for relational similarity: set resemblance of neighbour tuples and random walk probability, and then analysed subtle linkages effectively. Han et al. investigated two supervised learning approaches to disambiguate authors in the citations [8]. In [9], Kalashnikov proposed a domain-independent method. This method analysed not only object features but also inter-object relationships to improve the disambiguation quality. They used the shortest path algorithm to find the connect path connects two entities. And the path is measured by the association strength between the two entities. However, it is difficult to set the association strength between two entities correctly.

RDF is the W3C standard model for describing metadata. The RDF data also has the problem of duplicates. RDF data do not only represent the value of the data, but also represent the relationships among the data. Thus some studies represent the RDF data as graph. In [3], Klyne et.al. proposed a directed labelled graphs to represent the RDF data. A triple of RDF statement is a label edges. This model is easy to implement and represents the relationships among data clearly. However, if the relationships are complex, this model may miss some information. The RDF graph is different from a common graph. It is a hypergraph, because there may have more than one edge between two nodes. Therefore, in [1], Hayes proposed a undirected hypergraph model. In this model, each RDF statement is a hyperedge in the hypergraph. This model can represent the complex relationships among data. However, it cannot deal with the scale of RDF data, and is hard to more process on it. And in [1], they also proposed a bipartite graph model. This model transfers the hypergraph to a common bipartite graph. It is easy to do more processes. Our model is improved from this model. All the models introduced are using for semantic retrieval, like similar query and related query. They do not force on the duplicate detection.

6 Conclusion and future works

Nowadays, more and more resources are stored as RDF data. However, seldom researchers study on RDF data cleansing, although the problem of duplicates still exists in RDF data. According to that, in this paper, we proposed an approach to detect the duplicates in RDF data.

The proposed approach combines both similarity and connection among RDF data to detect the duplicates. And considering the complex of the association among RDF data, we proposed a new model for RDF, called RDF-Bipartite graph. This model is improved from Bipartite Statement-Value Graphs [1]. And we also proposed a method to find connect-path between two nodes in the RDF-Bipartite graph, called Subgraph-Extending method. This method does not find the connect-path directly, and instead to finding the subgraph contains both nodes. And introduce the connect resistance to help picking up the stronger connection pairs. This method is easy and efficiently to find the connection between two nodes.

We experiment the proposed method on publication datasets, and compare it with the traditional method. The results show that the proposed method improves accuracy and efficiency in detecting duplicates obviously.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61100055), The major projects of the National Social Science Fundation of China(Grant Nos.11&ZD189), Natural Science Foundation of Hubei Province (Grant No.500104), the Hubei Province Key Laboratory of Intelligent information processing and real-time industrial systems (Wuhan University of Science and Technology, Grant Nos. znss2013B013).

References

[1] Amadis A, Gutierrez C 2007 A Directed Hypergraph Model for RDF *In proceeding of: Proceedings of the KWEPSY 2007 Knowledge Web PhD Symposium 2007*, Innsbruck, Austria, June 6, 2007 pp. 47–61

[2] Huang L, Jin H, Yuan P, Chu F 2008 Duplicate Records Cleansing with Length Filtering and Dynamic Weighting *International Conference on Semantics, Knowledge and Grid 2008, Dec. 4-6, 2008* 95-102

[3] Minton S, Nanjo C 2011 A Heterogeneous Field Matching Method for Record Linkage *In: Proceedings of the Fifth IEEE International Conference on Data Mining* 314-321

[4] Atsuko Y, Yasunori Y, Kim In-Dong 2012 *Discriminative application of string similarity methods to chemical and non-chemical names for biomedical abbreviation clustering* BMC Genomics **13**(3) S8

[5] Bunke H 2012 On the Weighted Mean of a Pair of Strings *Pattern Analysis & Applications* **5** 23-30

[6] Anderson A 2010 Effective self-training author name disambiguation in scholarly digital libraries *Proceedings of the 10th annual joint conference on Digital libraries* 39-48

[7] Yin X, Han J, Hu P 2010 Object Distinction: Distinguishing Entities with Identical Names *In ICDE 2010: IEEE 23rd International Conference* 1242-1246
[8] Han H, Giles L, Zha H 2012 Two Supervised Learning Approaches for Name Disambiguation in Author Citations *JCDL'12* 269-305
[9] Kalashnikov D, Mehrotra S 2012 Domain-independent data cleaning via analysis of entity-relationship graph *ACM Transaction on Database Systems* **31**(2) 716-767
[10] Klyne G, Carroll J 2004 *Resource description framework (RDF): Concepts and Abstract Syntax* W3C Recommendation http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ 10 February

Authors

Huang Li, born in 1982, Wuhan, China

Current position, grades: lecturer of computer science
University studies: PhD in computer science
Scientific interest: Data management, Semantic web and knowledge.
Publications: 10
Experience: Li Huang is a PhD in computer science. Currently, she is a lecturer of computer science of the School of Computer Science and Technology, WUST, Wuhan, China. Her research interests include data management, semantic web and knowledge.