njuLink: Results for Instance Matching at OAEI 2017

Xinze Lyu, Qingheng Zhang, Wei Hu(✉), Zequn Sun, and Yuzhong Qu

[1] State Key Laboratory for Novel Software Technology, Nanjing University, China
[2] Department of Computer Science and Technology, Nanjing University, China
{xzlv.nju,qhzhang.nju,zqsun.nju}@gmail.com,{whu,yzqu}@nju.edu.cn

Abstract. njuLink is a tool designed for instance matching. It mainly matches instances by finding discriminative property pairs. Also, to meet 1:1 equivalence relationship for the OAEI 2017 DORUMES task, we make several improvements. In this report, we describe the design ideas and show our evaluation results.

1 Presentation of the System

1.1 State, purpose, general statement

With the rapid development of the Semantic Web, the amount of RDF data on the Semantic Web is growing in an unprecedented pace. This also brings great challenges to instance matching. On the Semantic Web, an instance describes a real-world object, it is composed of a subject and many $\langle p, v\rangle$ pairs, where p denotes a "property" and v denotes a "value". Subject serves as unique token for a real-world object, and $\langle p, v\rangle$ pairs describe the features of this real-world object. Instance matching aims to find the instances that describe the same real-world object and establish links between them. If two instances describe the same real-world object, we consider them as *coreferent instances* or a *coreferent instance pair*. Thanks to a lot of existing work, e.g., the Linked Open Data (LOD) Initiative, millions of links have been established. But, there are still a huge number of instances that potentially refer to the same object but have not been interlinked yet.

Our previous work tries to find coreferent instances by discriminative properties [2]. This approach is very effective but needs some improvements to meet the requirements of the DOREMUS task, which is to find 1:1 equivalence relationship between two datasets. So, we design njuLink, where "nju" represents "Nanjing University". The key idea of njuLink lies in finding what is essential to determine whether two instances are coreferent. Driven by this, first, njuLink builds a small-scale training set via predicting coreferent and non-coreferent instance pairs. Then, by analyzing the value similarity of every instance pair in training set, njuLink finds some property pairs named *discriminative property pairs*, which have the ability to identify whether two instances are coreferent. Finally, for an instance pair, njuLink calculates the similarity of values based

on the discriminative property pairs, the similarity of values based on common property pairs and the similarity of properties that they have to determine if the instances in this pair is coreferent.

1.2 Specific techniques used

There are four steps in the workflow of njuLink, which is shown in Fig. 1. We will describe the strategies to calculate the similarity of values and the similarity of properties shortly.

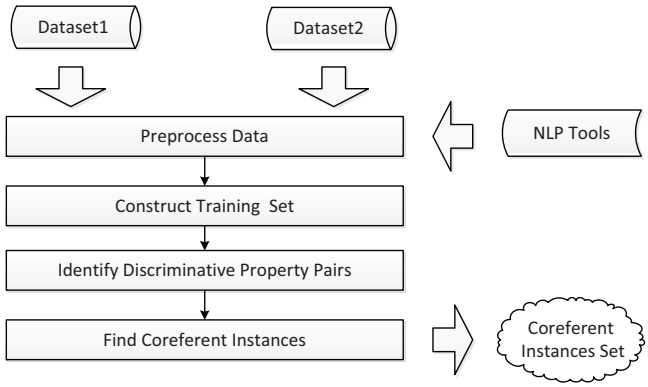

Fig. 1. The work flow of njuLink

The task we participated in is to find coreferent instance pairs between two datasets. To make our descriptions more clear, we give some notations as follows: (1) Let D^x and D^y be two different datasets, respectively; (2) The elements with superscript x are from D^x and those with superscript y are from D^y, e.g., instances, properties and values in D^x are i^x, p^x and v^x, respectively; and (3) Every instance pair $\langle i^x, i^y \rangle$ mentioned in this article is composed of an instance i^x from D^x and an instance i^y from D^y, and i^x is written to the left and i^y is written to the right, this also applies to property pairs $\langle p^x, p^y \rangle$ and value pairs $\langle v^x, v^y \rangle$.

Preprocess Data. For an instance, njuLink preprocesses the values describing it. There are three types of values: Blank node, URI and Literal (plain or typed). If a value is blank node, njuLink ignores it. Literal is divided into two kinds: typed literal, like boolean and integer, and plain literal, which is often accompanied with a language tag.

First, njuLink records the type of each value. Then, if the value has a language tag, njuLink also records it. Thirdly, for literals, njuLink replaces punctuations and stop words like “at”, “in”, “for” with space by a NLP tool, and then njuLink removes all space. For URIs, njuLink only records its local name. Finally, njuLink

transforms subjects, properties and values to lowercase letters and stores them for the next step.

Strategies to Calculate Similarity. We describe our strategies to obtain the similarity of a value pair and the similarity of a property pair next.

Calculate similarity of a value pair. Let v^x and v^y be two values owned by properties p^x and p^y, respectively. First, njuLink judges whether v^x and v^y are meaningful to be compared. There are three situations under which comparing them are not meaningful: (1) They both have language tags and their language tags are different; (2) The types of them are different; and (3) One of them is blank node.

Second, let $T(v^x)$ be the type of v^x. If v^x and v^y are meaningful to be compared, the strategies to find their similarity, denoted by $ValSim(v^x, v^y \mid p^x, p^y)$, vary with their types:

$$ValSim(v^x, v^y \mid p^x, p^y) = \begin{cases} indicatorFunc(v^x, v^y), & T(v^x) = \text{typed literal} \\ \text{I-Sub}(v^x, v^y), & \text{otherwise} \end{cases} \quad (1)$$

where for typed literal, njuLink uses indicator function ($indicatorFunc(v^x, v^y)$) to get their similarity, e.g., when two literals are both date time type, their similarity is 1 if the two literals are equal, and 0 otherwise. For URI and plain literal, njuLink uses I-Sub [3] to calculate the similarity. When the similarity of v^x and v^y is higher than a threshold, they are considered as a similar value pair. The threshold is set to 0.65, which is suggested by the authors of I-Sub [3].

Calculate similarity of a property pair. Let p^x and p^y be two properties owned by instances i^x and i^y, respectively. A property may have more than one value, we let the sets of values of p^x and p^y be $Val(p^x, i^x)$ and $Val(p^y, i^y)$, respectively. First, we find value set that has a smaller size. Without loss of generality, we assume that $Val(p^x, i^x)$ is the smaller one here. For a value v^x in $Val(p^x, i^x)$, the maximum similarity between it and the values in $Val(p^y, i^y)$ is calculated by $MaxValSim(v^x, Val(p^y, i^y))$. The maximum similarity between values of $Val(p^x, i^x)$ and $Val(p^y, i^y)$, which is also considered as the maximum similarity of property pair $\langle p^x, p^y \rangle$, is denoted by $MaxPropSim(p^x, p^y \mid i^x, i^y)$:

$$MaxValSim(v^x, Val(p^y, i^y)) = \max_{v_n^y \in Val(p^y, i^y)} ValSim(v^x, v_n^y \mid p^x, p^y), \quad (2)$$

$$MaxPropSim(p^x, p^y \mid i^x, i^y) = \max_{\substack{v_m^x \\ \in Val(p^x, i^x)}} MaxValSim(v_m^x, Val(p^y, i^y)). \quad (3)$$

If $MaxValSim(v^x, Val(p^x, i^x))$ of v^x is higher than a threshold (i.e. 0.65), value v^x is considered as a matched value, we define the sets of matched values and unmatched values between p^x of i^x and p^y of i^y as follows:

$$MatVal(p^x, p^y \mid i^x, i^y) = \{v \mid v \in Val(p^x, i^x) \cap MaxValSim(v, Val(p^y, i^y)) > 0.65)\}, \quad (4)$$

$$UnmatVal(p^x, p^y \mid i^x, i^y) = \{v \mid v \in Val(p^x, i^x) \cap v \notin MatVal(p^x, p^y \mid i^x, i^y)\}. \quad (5)$$

If $MaxPropSim(p^x, p^y \mid i^x, i^y)$ is higher than a threshold (0.65), the property pair $\langle p^x, p^y \rangle$ is similar w.r.t. instance pair $\langle i^x, i^y \rangle$. Note that this property pair is not guaranteed to be similar in another instance pair. For every matched value v^x of $Val(p^x, i^x)$, we sum up its similarity by $MatValSimSum(p^x, p^y \mid i^x, i^y)$:

$$MatValSimSum(p^x, p^y \mid i^x, i^y) = \sum_{\substack{v_m^x \in MatVal(\\ p^x, p^y \mid i^x, i^y)}} MaxValSim(v_m^x, Val(p^y, i^y)). \tag{6}$$

Construct Training Set. Let D^x and D^y be two different datasets and $\langle i_m^x, i_n^y \rangle$ be an instance pair, where i_m^x is from D^x and i_n^y is from D^y. The training set is divided into two parts, Positives and Negatives. Positives consist of coreferent instance pairs and Negatives are composed of non-coreferent instance pairs.

To construct Positives, njuLink picks up 20 instance pairs that have at least one property pair whose maximum similarity is very high. The threshold of similarity under this situation is 1.

When it comes to Negatives, njuLink chooses 20 instances from D^y randomly to form an instance set, namely instSet^y. These 20 instances should be under the same class of i_n^y in Positives, i.e., if instances in Positives are to describe "student", the instances selected should describe "student", too.

Then, njuLink picks up instances i_m^x from every instance pair $\langle i_m^x, i_n^y \rangle$ in Positives to form another instance set, namely instSet^x. So, instSet^x contains 20 instances because there are 20 instance pairs in Positives. After that, for every one in instSet^x, njuLink selects an instance from instSet^y and makes them an instance pair. Note that every instance in instSet^x and instSet^y is used only once. Finally, 20 generated instance pairs constitute the Negatives.

These 20 generated instance pairs can be considered as non-conferent ones approximately because the number of non-coreferent instances is much more than that of coreferent instances and njuLink constitutes instSet^y by selecting instances randomly.

Identify Discriminative Property Pairs. For every instance pair $\langle i_m^x, i_n^y \rangle$ from Positives, where i_m^x and i_n^y represent two different instances, njuLink makes every property of i_m^x and every property of i_n^y a pair. Then, njuLink finds out which property pair is similar and records it. So, njuLink can get the frequency of every similar property pair recorded after checking all instance pairs. If the frequency of a property pair is more than half of the size of Positives, which equals 10 in this case, njuLink records it in candidate property pair set.

For every property pair $\langle p_k^x, p_j^y \rangle$ in candidate property pair set, where p_k^x and p_j^y represent properties, njuLink calculates the maximum similarity that an instance pair $\langle i^x, i^y \rangle$ on it ($MaxPropSim(p_k^x, p_j^y \mid i^x, i^y)$). If the similarity is higher than a threshold, which is 0.65, this instance pair is a coreferent instance pair found by $\langle p_k^x, p_j^y \rangle$, otherwise, this instance pair is not coreferent judged by $\langle p_k^x, p_j^y \rangle$.

The percentage of the number of coreferent instances found can measure the discriminability of a property pair, but we found a better approach in [1] to use *information gain*, which is widely used in classification.

Every property pair $\langle p_k^x, p_j^y \rangle$ of candidate property pair set can classify the whole training set to four sets, TP, FP, TN and FN, which denote true positives, false positives, true negatives and false negatives respectively. When an instance pair is coreferent, if it is also a coreferent one found by $\langle p_k^x, p_j^y \rangle$, it is put into TP, otherwise, it is put into FP. When an instance pair is not coreferent, if it is also a non-coreferent one judged by $\langle p_k^x, p_j^y \rangle$, it is put into TN, otherwise, it is put into FN.

Finally, let T be the training set, which is the union of Positives (T^+) and Negatives (T^-). For every property pair $\langle p_k^x, p_j^y \rangle$ in candidate property pair set, njuLink uses four sets generated by it to obtain the *information gain* of it, denoted by $IG(p_k^x, p_j^y)$:

$$IG(p_k^x, p_j^y) = E(T) - E(T_{\langle p_k^x, p_j^y \rangle}), \tag{7}$$

$$E(T) = \frac{|T^+|}{|T|} \log \frac{|T^+|}{|T|} - \frac{|T^-|}{|T|} \log \frac{|T^-|}{|T|}, \tag{8}$$

$$E(T_{\langle p_k^x, p_j^y \rangle}) = \frac{|P|}{|T|} E(P) - \frac{|Q|}{|T|} E(Q), \tag{9}$$

$$E(P) = \frac{|TP|}{|P|} \log \frac{|TP|}{|P|} - \frac{|FN|}{|P|} \log \frac{|FN|}{|P|}, \tag{10}$$

$$E(Q) = \frac{|FP|}{|Q|} \log \frac{|FP|}{|Q|} - \frac{|TN|}{|Q|} \log \frac{|TN|}{|Q|}, \tag{11}$$

$$P = TP + FN, \tag{12}$$

$$Q = FP + TN, \tag{13}$$

where $E(T)$ measures the information entropy of the original training set T, and $E(T_{\langle p_k^x, p_j^y \rangle})$ measures the information entropy after using $\langle p_k^x, p_j^y \rangle$ to classify instance pairs in T. If $IG(p_k^x, p_j^y)$ is higher than a threshold, $\langle p_k^x, p_j^y \rangle$ is considered as a *discriminative property pair*. We set the threshold 0.2 in our tool. njuLink gets a set of discriminative property pairs after checking all property pairs in candidate property pair set.

Find Coreferent Instances. The key ideas to find coreferent instances are from two aspects: (1) Get detailed similarity w.r.t. an instance pair; and (2) Find the most coreferent instance pair, e.g., for an instance i and an instance set $instSet$, we assume that every instance in $instSet$ seems to be coreferent with i. To find the real coreferent instance pair, first, we use every instance in $instSet$ to form an instance pair with i, and then, we compare the detailed similarity of each instance pair formed and only record the instance pair with highest similarity. It guarantees 1:1 equivalence relationship between two datasets.

Let $DiscrPropSet(D^x, D^y)$ denote the set of discriminative property pairs. First, for every instance in D^x, njuLink combines it with every instance in D^y

to generate many instance pairs, and for every generated instance pair $\langle i_m^x, i_n^y \rangle$, njuLink finds the set of similar discriminative property pair for it, which is denoted by $SimDiscrPropSet(i_m^x, i_n^y)$:

$$\begin{aligned} SimDiscrPropSet(i_m^x, i_n^y) = &\{\langle p^x, p^y \rangle \mid \langle p^x, p^y \rangle \in DiscrPropSet(D^x, D^y) \\ &\& \; p^x \in Prop(i_m^x) \; \& \; p^y \in Prop(i_n^y) \\ &\& \; MaxPropSim(p^x, p^y \mid i_m^x, i_n^y) > 0.65\}. \end{aligned} \tag{14}$$

where $Prop(i_m^x)$ and $Prop(i_n^x)$ are the sets of properties of i_m^x and i_n^y, respectively. Secondly, njuLink calculates seven features below to represent the similarity of the pair:

1) The size of $SimDiscrPropSet(i_m^x, i_n^y)$.
2) The sum of information gain of each similar discriminative property pair $IGSum(i_m^x, i_n^y)$:

$$IGSum(i_m^x, i_n^y) = \sum_{\langle p_k^x, p_j^y \rangle \in SimDiscrPropSet(i_m^x, i_n^y)} IG(p_k^x, p_j^y), \tag{15}$$

3) The sum of detailed information gain of each similar discriminative property pair $DIGSum(i_m^x, i_n^y)$:

$$DIGSum(i_m^x, i_n^y) = \sum_{\substack{\langle p_k^x, p_j^y \rangle \\ \in SimDiscrPropSet(i_m^x, i_n^y)}} DIG(p_k^x, p_j^y \mid i_m^x, i_n^y), \tag{16}$$

$$\begin{aligned} DIG(p_k^x, p_j^y \mid i_m^x, i_n^y) = &(|MatVal(p_k^x, p_j^y \mid i_m^x, i_n^y)| \\ &- |UnmatVal(p_k^x, p_j^y \mid i_m^x, i_n^y)|) * IG(p_k^x, p_j^y), \end{aligned} \tag{17}$$

where $DIG(p_k^x, p_j^y \mid i_m^x, i_n^y)$ denotes the detailed information gain of a similar discriminative property pair w.r.t. $\langle i_m^x, i_n^y \rangle$.

4) The sum of detailed similarity sum of each similar discriminative property $DSimSum(i_m^x, i_n^y)$:

$$DSimSum(i_m^x, i_n^y) = \sum_{\substack{\langle p_k^x, p_j^y \rangle \\ \in SimDiscrPropSet(i_m^x, i_n^y)}} DSim(p_k^x, p_j^y \mid i_m^x, i_n^y), \tag{18}$$

$$\begin{aligned} DSim(p_k^x, p_j^y \mid i_m^x, i_n^y) = &MatValSimSum(p_k^x, p_j^y \mid i_m^x, i_n^y) \\ &* IG(p_k^x, p_j^y), \end{aligned} \tag{19}$$

where $DSim(p_k^x, p_j^y \mid i_m^x, i_n^y)$ denotes the detailed similarity sum of a similar discriminative property w.r.t. $\langle i_m^x, i_n^y \rangle$.

5) The number of similar common property pairs.
6) The sum of maximum similarity of each similar common property pair w.r.t. $\langle i_m^x, i_n^y \rangle$.

7) The number of property pairs that two properties of each one have the same local names. We make every property in $Prop(i_m^x)$ and every property in $Prop(i_n^y)$ a property pair and check them all.

Besides discriminative property pairs, we also use three features from common property pairs because we find discriminative property pairs are not enough to separate the most coreferent instance pairs from those that seem to be coreferent. A common property pair should meet two requirements: this property pair is not a discriminative property pair and two properties of it have the same local names.

Thirdly, njuLink sorts the instance pairs generated in descending order according to these seven scores of each one. The importance of these seven features is 1) > 2) > 3) > 4) > 5) > 6) > 7). Finally, njuLink selects instances in sorted instance pairs set from top to bottom, meanwhile, when we pick up instance pairs from top to bottom, if two instances of an instance pair are both the first time to be checked, we record it, otherwise, drop it. It guarantees the 1:1 equivalence relationship between two datasets D^x and D^y.

1.3 Link to the system and parameters file

You can find the source code and the jar tested by SEALS client successfully on GitHub: `https://github.com/nju-websoft/njuLink`.

1.4 Link to the set of provided alignments (in align format)

The alignment files for DOREMUS task should be available at the official website: `http://islab.di.unimi.it/content/im_oaei/2017/`.

2 Results for DOREMUS

There are two sub-tasks under DOREMUS, namely HT and FPT. HT aims to obtain 1:1 equivalence relationship between instances whose data have different types of heterogeneities, while FPT aims to get the same relationship as that of HT between instances with high similarity.

njuLink succeeds in finding property pairs with high discriminability, which are shown in Table 1. The results of evaluation are shown in Table 2 and Table 3.

3 Discussions about improvements

How to apply different approaches according to different datasets automatically? During the development of njuLink, we adjust the way to find coreferent instances according to the requirements of DOREMUS. But the adjusted approach is not applicable for all tasks. So, finding a way to decide appropriate approaches automatically is necessary.

Table 1. Discriminative property pairs on the DOREMUS task

	Properties in dataset 1	Properties in dataset 2
HT	U70_has_title	U70_has_title
	U70_has_title	label
	label	label
	label	U70_has_title
	U16_has_catalogue_statement	U16_has_catalogue_statement
FPT	U70_has_title	U70_has_title
	U70_has_title	label
	label	label
	label	U70_has_title

Table 2. Results for HT

	Precision	Recall	F1-score
AML	0.851	0.479	0.613
I-Match	0.680	0.071	0.129
Legato	0.930	0.920	0.930
LogMap	0.406	0.882	0.556
njuLink	0.966	0.945	0.955

Table 3. Results for FPT

	Precision	Recall	F1-score
AML	0.914	0.427	0.582
I-Match	1.000	0.053	0.101
Legato	1.000	0.980	0.990
LogMap	0.119	0.880	0.210
njuLink	0.959	0.933	0.946

4 Conclusion

njuLink is dedicated to finding coreferent instances by utilizing discriminative property pairs. The Instance Matching track of this year show many new things to us. This helps us find the weaknesses of njuLink and makes our original ideas better. Technical problems happened during the development also forced us to pay more attention to the way of realizing our tool. In the future, we will continue following the trends of instance matching with interests and try to solve issues on which we have not achieved good performance.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 61370019). During our development, we received much support from organizers and volunteers of OAEI, we would like to thank them for their help.

References

1. Hu, W., Jia, C.: A bootstrapping approach to entity linkage on the semantic web. Journal of Web Semantics 34, 1–12 (2015)
2. Hu, W., Yang, R., Qu, Y.: Automatically generating data linkages using class-based discriminative properties. Data & Knowledge Engineering 91, 34–51 (2014)
3. Stoilos, G., Stamou, G., Kollias, S.: A string metric for ontology alignment. In: ISWC 2005. pp. 624–637. Springer (2005)