IEICE TRANSACTIONS

on Information and Systems

VOL. E99-D NO. 6
JUNE 2016

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER

Linked Data Entity Resolution System Enhanced by Configuration Learning Algorithm

Khai NGUYEN[†,††a)], ***Nonmember and*** **Ryutaro ICHISE**[†,††b)], ***Member***

SUMMARY Linked data entity resolution is the detection of instances that reside in different repositories but co-describe the same topic. The quality of the resolution result depends on the appropriateness of the configuration, including the selected matching properties and the similarity measures. Because such configuration details are currently set differently across domains and repositories, a general resolution approach for every repository is necessary. In this paper, we present *cℒink*, a system that can perform entity resolution on any input effectively by using a learning algorithm to find the optimal configuration. Experiments show that *cℒink* achieves high performance even when being given only a small amount of training data. *cℒink* also outperforms recent systems, including the ones that use the supervised learning approach.

key words: *linked data, entity resolution, schema-independent, supervised, heuristic*

1. Introduction

Finding co-referent instances, which at the same time describe the same topic, is an important process in data integration. Because data is created independently in different repositories, gathering data from various sources greatly enriches the information. In linked data publication, making co-referent links between the newly published instances and the existing ones in the web of linked data is an indispensable step. The problem addressed by entity resolution is finding all co-referent instances residing in two different repositories: the source and the target. Entity resolution has been a subject of extensive research [1], [2]; however, finding the perfect resolution algorithm remains a work in progress.

The challenge of entity resolution is the difference between the representation of information in the source and the target repositories. Although such difficulty also exists in other sorts of data (e.g., relational database), for linked data and other web-based data, the open access mechanism increases the difficulty. In linked data, besides the difference in values (e.g., many names for one place), the difference in schemata is also a big issue. Since the schema specifies how the details of instances are declared, for different domains or repositories have different schema. Therefore, an entity resolution system has to use an appropriate matching configuration designed particularly for the input repositories. For example, a suitable configuration for two repositories of 'people' domain specifies two comparisons: on 'name' and 'birth date'. However, for other repositories, with other domain (e.g., 'place' and 'organization') or even with the same domain 'people' but different in schema (e.g. 'name' is replaced by 'label'), such configuration is no longer appropriate. Therefore, it is important to develop a system that can adapt to any repository with any schema. Such a system is called schema-independent.

A schema-independent system finds the matching configuration itself using the observation on the input repositories. Many techniques have been used, such as pure statistic [3], [4], unsupervised learning [5], and supervised learning [6]–[11]. Supervised learning requires a number of labeled co-referent instances but this requirement is compensated by the achievement of the best accuracy compared to other techniques. Recently, supervised learning of configuration has been investigated with genetic algorithm [8], [9] and information-gain based selection [7]. The reported results of using these methods are promising. Unfortunately, the genetic algorithm is not supported by a clear strategy as it is based on random search principle. Meanwhile, the information-gain based selection ignores the evaluation of combining different similarity estimation methods. Therefore, the current achievements of previous supervised systems can be further improved.

We focus on the problem of entity resolution using the supervised learning principle. We describe *cℒink*, a schema-independent entity resolution system, which is enhanced by a novel supervised configuration learning algorithm. This algorithm is more effective than other algorithms having the same objective. Given two input repositories, *cℒink* performs a pre-learning stage to generate co-referent candidates. Some candidates are labeled with positive and negative labels and input to the learning algorithm, which uses a heuristic search method to optimize the combination of similarity functions. The learning outcome is an optimal configuration that is effective for discovering the co-references from unlabeled candidates. This paper is the complete version of [12], in which *cℒink* is first proposed. Here we provide more detailed contents as well as important experiments.

The remainder of this paper is organized as follows: Sect. 2 discusses related work. Section 3 describes the details of *cℒink*. Section 4 reports the experiments. Section 5

Manuscript received September 28, 2015.
Manuscript revised January 26, 2016.
Manuscript publicized February 29, 2016.

[†]The authors are with the Graduate University for Advanced Studies, Shonan Village, Kanagawa-ken, 240–0193 Japan.
[††]The authors are with National Institute of Informatics, Tokyo, 101–8430 Japan.
a) E-mail: nhkhai@nii.ac.jp
b) E-mail: ichise@nii.ac.jp

DOI: 10.1587/transinf.2015EDP7392

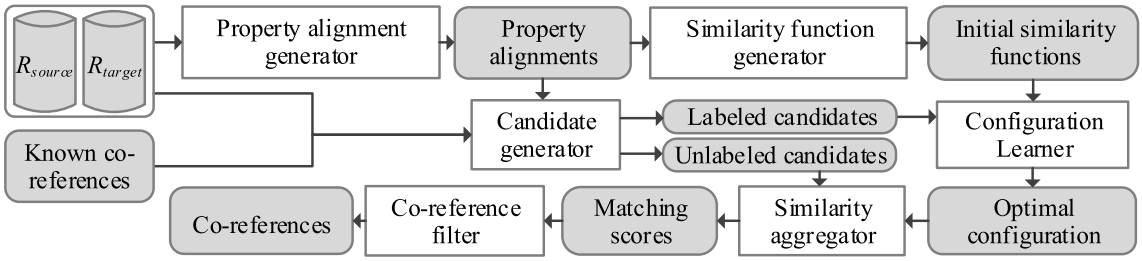

Fig. 1 The workflow of *cLink*.

summarizes the paper and future work.

2. Related Work

In this section, we review the work focusing on linked data entity resolution. The early stages of entity resolution are the systems whose configurations are constructed by user. SILK [13] is among the pioneers of linked data interlinking frameworks. SILK provides a declarative structure to represent a user-defined matching configuration. Focusing on efficiency aspect, LIMES [14] and ScSLINT [15] have a significant improvement in processing time. *cLink* is also implemented as a part of ScSLINT framework. AggrementMaker [16], RiMOM [17], and Zhishi.Links [18] aimed at accuracy and scalability. Zhishi.Links includes domain knowledge to improve matching accuracy. AggrementMaker and RiMOM combine different matching strategies to leverage multitude of information. RiMOM is also one of the state-of-art entity resolution systems for linked data.

As the application of manual approach is limited due to the diversity of domains and schemata, a large number of automatic approaches have been proposed. SERIMI [3] and SLINT+ [4] attempted to eliminate user involvement by automatically detecting property alignments using overlap measure. KnoFuss [5] applies genetic algorithm to find an optimal matching configuration. The fitness is represented by a pseudo value of actual accuracy. One advantage of these systems is that the existing co-references is not required. However, by not using such labeled data, the quality of the detected alignments cannot be correctly evaluated. In other words, the obtained configuration can contain incorrect alignments and thus reduces the performance.

Recently, the supervised learning approach has become practical because of the abundance of existing co-references in linked data. ADL [7] and ObjectCoref [6] learn discriminative properties for two repositories having the same class. Rong et al. used Adaboost to train a binary classifier, which can determine whether a pair of instances is co-referent or not [11]. The disadvantage of using classifier is that the similarities of instances are not explicitly defined and thus a system cannot perform further tasks that need the similarities (e.g., post-filtering in SLINT+ improves the resolution result). Most related to our work are RAVEN [9], EAGLE [10], and ActiveGenLink [8]. These systems use active learning to find the optimal link specification, whose core component is the matching configuration. EAGLE and ActiveGenLink apply genetic algorithm to improve the efficiency. Genetic algorithm has advantage in learning on data with many property alignments. However, genetic algorithm has to check many configurations to reach the convergence.

Other remarkable systems are PARIS [19], Markov logic-based matcher [20], and ZenCrowd [21]. PARIS and Markov logic-based matcher are the automatic systems focusing on probabilistic methods. Zencrowd combines an automatic algorithm and crowdsourcing. Although these systems obtained good results in experiment, when the high accuracy is the first priority and the existing co-references are available, the supervised learning approach is still the first option.

3. The *cLink* System

cLink is built upon the general architecture of ScSLINT framework [15]. *cLink* consists of six components, property alignment generator, similarity function generator, candidate generator, configuration learner, similarity aggregator, and co-reference filter. The workflow of *cLink* is illustrated in Fig. 1. Given two input repositories, R_{source} and R_{target}, *cLink* first creates the property alignments. Then, these alignments are used to build the initial similarity functions and to select the possibly co-referent candidates. Using the known co-references, some candidates are labeled and input into the configuration learner to find the optimal configuration. Optimal configuration defines the appropriate similarity functions selected from the initial similarity functions. Each similarity function computes the similarity of two instances on one property. The similarity aggregator combines many similarities into a final matching score. Finally, co-references filter considers the matching score of all candidates and produces the final co-references. Next, we describe the details of each component.

3.1 Property Alignment Generator

This component creates the property alignments between R_{source} and R_{target}. A property alignment is expected to describe the same attribute of two instances. In linked data, as properties are represented by RDF predicates, the outputs of this component are the alignments between RDF predicates.

Property alignments are generated in two steps. First, *cLink* selects in the source repository the predicates that satisfy two conditions. Second, *cLink* aligns each selected

predicate with the corresponding ones in the target repository using an overlap measure. The conditions used in the first step are the discriminative and the coverage, which are customized from [22]. The discriminability $dis(p_k)$ expresses the diversity of the RDF objects declared by p_k, while the coverage $cov(p_k)$ represents the instance-wise frequency of p_k. Equation (1) and (2) describe $dis(p_k)$ and $cov(p_k)$, respectively:

$$dis(p_k) = \frac{|\{o|x \in R_{source}, < s, p_k, o > \in x\}|}{|\{< s, p_k, o > |x \in R_{source}, < s, p_k, o > \in x\}|} \quad (1)$$

$$cov(p_k) = \frac{|\{x|x \in R_{source}, < s, p_k, o > \in x\}|}{|R_{source}|} \quad (2)$$

where R stands for repository, $< s, p, o >$ is a RDF triple, and x is an instance, which is a set of RDF triples sharing the same subject s. We separate the predicates by their data type: *string*, *number*, *date*, and *URI*. The type of a predicate is assigned with the most frequent type of its RDF objects. For each type, we select the predicates having the discriminability dis higher than α and then retain the K_{cov} predicates with the highest coverages cov.

One predicate selected from the source repository can be aligned with many predicates having the same type in the target. The confidence $conf$ of an alignment $[p_{source}, p_{target}]$ is measured as follows:

$$conf([p_{source}, p_{target}]) = \frac{|O_{p_{source}} \cap O_{p_{target}}|}{|O_{p_{source}}|} \quad (3)$$

$$O_{p_k} = \{E(o)|x \in R_k, < s, p_k, o > \in x\}$$

where E is a pre-processing function. E is applied for each RDF object. E works in flexible ways depending on the property type. For *string*, E collects the tokens. For *number*, E returns the rounded value at two decimal digits. For *date*, E keeps the original value. For *URI*, E extracts the remaining string after stripping away the domain. For each predicate of the source repository, we select the top K_{align} alignments having the highest $conf$. In total, this step produces at most $K_{cov} \times K_{align}$ property alignments for each type. In technical aspect, a very small threshold is applied to $conf$ to avoid accepting slightly related property alignments. We set this threshold into 0.01 by default.

Equation (3) works under the assumption that the target repository contains many co-references with the source repository. Therefore, the denominator is related only to the source repository instead of both repositories like a Jaccard measure, which is used in SERIMI [3] and SLINT+ [4]. Equation (3) is reasonable because currently there are many large repositories that cover a wide range of instances (e.g., DBpedia, Freebase, and Wikidata). Furthermore, as the predicate alignments are generated using only RDF objects, *cLink* does not require the specification of the schemata. In other words, *cLink* is schema-independent.

3.2 Similarity Function Generator

This component uses the property alignments generated by the previous component to create the prototype[†] of the initial set of similarity functions. This set is later used to select the similarity functions for the optimal configuration. Each similarity function computes the similarity of two instances on one designated property. Equation (4) is the definition of a similarity function sim:

$$sim_{\{[p_{source}, p_{target}], sm\}}(x, y) = \max_{o_x, o_y}(sm(E(o_x), E(o_y))) \quad (4)$$
$$< s, p_{source}, o_x > \in x, < s, p_{target}, o_y > \in y$$

A similarity function is specified by two pieces of information: a property alignment $[p_{source}, p_{target}]$ and a similarity measure sm. Therefore, given multiple property alignments and similarity measures, different similarity functions are created. For two instances: $x \in R_{source}$ and $y \in R_{target}$, a similarity function returns the similarity of the most similar RDF objects declared by p_{source} and p_{target}. In other words, the max operator effects if p_{source} or p_{target} appears many times in x or y. The function E defined in Sect. 3.1 is used to pre-process the RDF objects.

The similarity measure sm is assigned to the similarity function in accordance with the type of p_{source} and p_{target}. *cLink* supports five similarity measures. For 'date' and 'URI', *cLink* uses the exact matching, which returns 1.0 if two values are identical and 0.0 otherwise. For 'number', *cLink* uses the reversed difference (Eq. (5)), which calculates how much close two numbers a and b are to each other:

$$rDiff(a, b) = (1 + |a - b|)^{-1} \quad (5)$$

For *string*, *cLink* supports many measures having different characteristics. For short string, we select Levenshtein because of its robustness. For long string, we use the well-known TFIDF-cosine in order to consider the overlap of token as well as the their weight.

The input of this component is all the generated property alignments. In parallel, the alignments of string properties are also input into the candidate generator, which is described in the following section.

3.3 Candidate Generator

Basically, every pair of instances between R_{source} and R_{target} need to be compared to find all the co-references. However, it is impractical to perform all pairwise comparisons, especially when the repositories are large. The mission of this component is to reduce the number of comparisons from $|R_{source}| \times |R_{target}|$ into a much smaller number. For that mission, we limit the comparisons to only the potentially co-referent instances, which are called candidates.

cLink finds the candidates by using a simple token-based prefix blocking approach. Using this approach, a pair of two instances is considered as a candidate if they share at least one first token of the RDF objects declared by any string property alignment. By using only the first token, *cLink* can retain as many correct candidates as possible.

[†]Only the declaration of similarity functions are created.

Algorithm 1: *cLearn*

```
   Input: Training set T, validation set V, list of similarity
   functions I_sim, list of similarity aggregators I_agg, integer K_top
   Output: Optimal configuration C_opt
1  C_agg ← ∅
2  foreach A ∈ I_agg do
3      visited ← ∅
4      foreach sim ∈ I_sim do
5          c ← Init(Agg ← A, F_sim ← sim, δ ← 0)
6          [c.δ, F1] ← EvaluateAndAssignThreshold(c, T)
7          c.σ_sim ← c.δ
8          visited ← visited ∪ {[c, F1]}
9      candidate ← TopHighestF1(visited, K_top)
10     while candidate ≠ ∅ do
11         next ← ∅
12         foreach g ∈ candidate do
13             foreach h ∈ candidate do
14                 c ← Init(Agg ← A, F_sim ←
                   g.c.F_sim ∪ h.c.F_sim, δ ← 0)
15                 [c.δ, F1] ←
                   EvaluateAndAssignThreshold(c, T)
16                 visited ← visited ∪ {[c, F1]}
17                 if g.c.F_sim ≠ h.c.F_sim and F1 ≥ g.F1 and
                   F1 ≥ h.F1 then
18                     next ← next ∪ {[c, F1]}
19         candidate ← next
20     F1 ← Evaluate(argmax_{v∈visited}(v.F1).c, V)
21     C_agg ← C_agg ∪ {[c, F1]}
22 [C_opt, F1] ← argmax_{v∈C_agg}(v.F1)
23 return C_opt
```

There are many studies on candidate generation that use weighting schemes to reduce the number of candidates [4], [23]. However, such reduction is companied with a drop in the number of correct candidates. For that reason, we use the simple token-based blocking without weighting.

Given some co-references as labeled data, the label of some candidates are assigned. If a candidate (x, y) exists in the input co-references, where $x \in R_{source}$ and $y \in R_{target}$, (x, y) is labeled as positive and all other candidates (x, z) are labeled as negative, where $y \neq z \in R_{target}$. Furthermore, the labeled candidates are divided into training set T and validation set V, which are used by the learning algorithm in the next step.

3.4 Configuration Learner

In this component, we first initialize the set of different similarity aggregators I_{agg}, which are later described in Sect. 3.5. The similarity aggregators I_{agg}, the labeled candidates (training set T and validation set V), and the initial similarity functions I_{sim} (Sect. 3.2) are input to the learning algorithm. The algorithm learns the optimal configuration C_{opt} that is most suitable to the input repositories. A configuration specifies the combination of similarity functions F_{sim}, the similarity aggregator Agg, the parameters δ_{sim} associated with each similarity function sim, and the parameter δ of the co-reference filter (Sect. 3.6).

We propose *cLearn*, whose initial stage is previously presented in [24]. *cLearn* uses a heuristic search method to optimize the combination of the similarity functions and the similarity aggregator. The pseudo code of *cLearn* is given in Algorithm 1. In this pseudo code, we use dot ('.') notation to indicate the member accessor. *Init* creates a configuration by assigning Agg, F_{sim}, and δ with given values. *Evaluate* first executes the similarity aggregator and the co-reference filter specified by a configuration, on a set of candidates. Then, based on the label of those candidates, it computes the performance, $F1$ score. $F1$ is the harmonic mean of the recall rec and the precision pre, which are calculated as follow:

$$rec = \frac{\text{Number of correctly detected co-references}}{\text{Number of actual co-references}} \quad (6)$$

$$pre = \frac{\text{Number of correctly detected co-references}}{\text{Number of all detected co-references}} \quad (7)$$

EvaluateAndAssignThreshold works similarly to *Evaluate*, but at the same time, it finds a value for the threshold $c.\delta$. After executing the similarity aggregator and co-reference filter, we first select the top N candidates from T with highest matching score, where N is equal to the number of actual co-references in T. The N selected candidates can contain both correct and incorrect co-references. Based on the label of the N selected candidates, we compute the performance $F1$. For $c.\delta$, we assign the lowest matching score of the correctly selected candidate. K_{top} is an integer value that controls the maximum quantity of similarity functions in the learned configuration. By default, K_{top} is set to 16. In addition, *visited* variable is used to count the number of checked configurations for each similarity aggregator A.

cLearn begins with the consideration of each single similarity function and then checks their combinations. This algorithm works with an underlying heuristic. It is the direct enhancement assumption (line 17). The performance of using a new combination must not be less than that of the components. This heuristic is reasonable as a list of similarity functions that reduces the performance has little possibility of generating a further list with improvement. In addition, this algorithm is generic because *EvaluateAndAssignThreshold* can be replaced by other functions having the same purpose. Therefore, this algorithm is compatible with any similarity-based matching system.

The validation set V (line 20) is important. Each iteration controlled by line 2 finds an optimal configuration with one similarity aggregator A. In other words, there are $|I_{agg}|$ configurations in C_{agg}. In order to select the most optimal one from C_{agg}, instead of just selecting the configuration having the best performance on T, we use V to increase the generality of the final configuration C_{opt}.

3.5 Similarity Aggregator

A similarity aggregator computes the final matching score for each candidate using the similarity functions F_{sim} and

their parameter δ_{sim} specified by a configuration. The computation of the matching score $mScore(x, y)$ for two instances x and y is defined as follows:

$$mScore(x, y) = weight(y) \times combine_{F_{sim}}(x, y) \quad (8)$$

where *weight* is a function weighting the target instance y and *combine* is a similarity combination function. *cLink* provides *non-weighting* and *weighting* versions. For *non-weighting*, $weight(y)$ simply returns 1.0. For *weighting*, the weight is calculated by Eq.(9):

$$weight(y) = \log_{\max_{t \in R_{target}} size(t)} size(y) \quad (9)$$

where $size(y)$ counts the number of RDF triples existing in y. By using Eq. (9), we assume that instances containing many triples are more prioritized. The logarithmic scale is used to reduce the weight of instances whose *size* is particularly large. This weighting method is effective when the target repository is very ambiguous, such as large repositories.

For similarity combination, *cLink* uses the following e1quation:

$$combine_{F_{sim}}(x, y) = \frac{1}{valid(U_{F_{sim}}(x, y))} \sum_{v \in U_{F_{sim}}(x,y)} v^k \quad (10)$$

$$U_{F_{sim}}(x, y) = \{sim(x, y) | sim(x, y) \geq \sigma_{sim}, sim \in F_{sim}\}$$

where $k \in \{1, 2\}$, *valid* is a counting function, and σ_{sim} is the parameter for each similarity function *sim*, which is determined automatically by *cLearn* (line 7). k controls the transformation for each similarity v. When $k = 1$, *combine* acts as a first order aggregation. When $k = 2$, we have a quadratic aggregation. There are also two variations of *valid*, which return the number of elements in $U_{F_{sim}}(x, y)$ and 1.0 always. The difference between these variations is that the latter penalizes the (x, y) pair having similarities $sim(x, y) < \sigma_{sim}$ while the former does not. In addition, *cLink* provides a *restriction* mechanism to enable or disable σ_{sim}. When disabling, all σ_{sim} are set to zero instead of their original value. In total, there are 16 combinations of *weight*, *valid*, k, and *restriction*. Consequently, there are 16 different aggregators supported by *cLink*. All aggregators are used to initialize I_{agg} in *cLearn* and let the configuration learner select the best one.

3.6 Co-Reference Filter

This component uses the matching scores of the candidates to construct the final co-references. In *cLink*, we reuse the adaptive filter implemented in SLINT+ [4]. This filter follows the idea of the stable marriage problem [25]. A pair of instances (x, y) is co-referent if its matching score $mScore(x, y)$, satisfies the conditional statement of Eq. (11):

$$mScore(x, y) \geq \max(\max_{z \in R_{source}} mScore(z, y), \max_{t \in R_{target}} mScore(x, t)) \quad (11)$$

where $x \in R_{source}$ and $y \in R_{target}$. This filtering strategy is specially effective on highly ambiguous data, when many candidates are very similar but only the most similar one is the expected co-references.

In addition, this component uses a cut-off filter to eliminate the incorrect candidates but satisfying Eq. (11). A threshold δ is used for this task. δ is assigned by the learning algorithm. Only instance pairs whose scores satisfy the condition statement of the filter and threshold δ are promoted to be a final co-reference.

Above we have described the details of *cLink*. The next section reports the experiments and the results.

4. Experiment

We report in total four experiments. The first experiment evaluates the candidate generator. The second experiment evaluates the effectiveness of the learning algorithm of *cLink*. The third experiment compares *cLink* with other systems, including the systems that use supervised learning approach. The fourth experiment analyzes the impact of the size of training data on the performance of *cLink*. In addition, we discuss the effectiveness of similarity functions and similarity aggregators. The details of the experiments are described from Sect. 4.3 to 4.7.

We implement *cLink* using C++ language and conduct all experiments on a FreeBSD computer equipped with two Intel E5-2690 CPUs and 256 GB memory. The source code of *cLink* and all datasets can be downloaded at http://ri-www.nii.ac.jp/ScSLINT.

4.1 Datasets

We use the standard real-world benchmarks provided by the entity resolution track of OAEI 2010 and OAEI 2012. The OAEI 2010 dataset contained five repositories related to healthcare domain: Sider, Diseasome, Drugbank, Dailymed, and DBpedia. The OAEI 2012 dataset contains four repositories: NYTimes (NYT), DBpedia, Freebase, and Geonames, with three domains: location (loc), organization (org) and people (peo). There are a few slight inconsistencies between the ground-truth provided by OAEI 2012 and our downloaded dump data, because of the difference in the release dates [†]. Therefore, we exclude 130 (0.298%) source instances which are related to such inconsistencies. The details of these subsets are given in Table 1.

These datasets are chosen because the purpose of this experiment is to know the performance of *cLink* on the real data with large size. Although there are few newer datasets provided by OAEI, they do not focus on real-world resolution problem, contain small size, or are designed for testing entity resolution systems with some special challenges (e.g., unusual string distortion, language translation). In addition, using these datasets offer comparisons between *cLink* and

[†]We use DBpedia 3.7, Freebase 2013/09/23, and Geonames 2014/02.

Table 1 Summary of OAEI 2010 (D1 to D5) and OAEI 2012 (D6 to D12) datasets

ID	Source repository				Target repository				Co-references
	Name	#Instances	#Properties	#RDF Triples	Name	#Instances	#Properties	#RDF Triples	
D1	Sider	2,670	10	96,269	Drugbank	19,689	118	507,495	1,142
D2	Sider	2,670	10	96,269	Diseasome	8,149	18	69,544	344
D3	Sider	2,670	10	96,269	Dailymed	10,002	27	131,064	3,225
D4	Sider	2,670	10	96,269	DBpedia	4,183,461	45,858	232,957,729	1,449
D5	Dailymed	10,002	27	131,064	DBpedia	4,183,461	45,858	232,957,729	2,454
D6	NYT loc-db	3,837	22	42,998	DBpedia	4,183,461	45,858	232,957,729	1,917
D7	NYT org-db	5,967	20	54,404	DBpedia	4,183,461	45,858	232,957,729	1,922
D8	NYT peo-db	9,944	20	103,341	DBpedia	4,183,461	45,858	232,957,729	4,964
D9	NYT loc-fr	3,840	22	43,037	Freebase	40,358,162	2,455,627	912,845,965	1,920
D10	NYT org-fr	6,045	20	59,111	Freebase	40,358,162	2,455,627	912,845,965	3,001
D11	NYT peo-fr	9,958	20	103,496	Freebase	40,358,162	2,455,627	912,845,965	4,979
D12	NYT loc-gn	3,785	22	42,302	Geonames	8,514,201	14	112,643,369	1,729

recent systems, which were tested on the same benchmarks.

4.2 Experimental Settings

4.2.1 Parameter Settings

For property alignment generator, we set $K_{cov} = 4$, and $K_{align} = 4$ uniformly for all property types so that the maximum number of property alignments are 64. We set K_{cov} and K_{align} to high values because we expect to get many similarity functions in order to know the system's capability of finding the optimal similarity functions combination. After that, we gradually reduce α from 1.0 until there is at least one string property alignment is selected for each subset. We observe that using $\alpha = 0.5$ satisfies such an expectation and use this value uniformly for all tests. We do not tune α for each subset because we are interested in testing the system using the same parameters for many different input.

Furthermore, varying α in the large ranges of [0.2, 0.7] and [0.1, 0.5] does not change the property alignments on OAEI 2010 and OAEI 2012 dataset, respectively. It implies that this component is not sensitive to parameters on tested real datasets and thus potentially remains the same advantage on other datasets.

4.2.2 Candidate Splitting

When splitting a set of candidates into smaller sets, we randomly separate the candidates with the constraint that all the candidates sharing the source instance are put together. In other words, each separated set does not share any source instance with each other.

For all experiments, we first split the candidates into two sets. The size of these sets is determined differently for each experiment. The first set is reserved for learning, and the second set (test set) is used for evaluating the performance. After that, the first set is split into training set T (80%) and validation set V (20%). Furthermore, since the source repository contains instances that are not co-referent with any instance in the target, an unmanaged randomization may not reflect the actual distribution of the data. Let r be the ratio between the number of the source instances having a co-reference and the ones that do not have any co-reference. We remain the ratio r for every separated set.

Our separation strategy is reasonable because it is compatible with the practical annotation process. Given an instance from source repository, a ranked list of instance pairs is created using a simple matching method. Annotators are asked to assign the positive labels for the top ranked pairs. Because the list is sorted, the remaining pairs can be assumed to be negative. By following this manner, such positive label assignments have tiny possibility to be incorrect. That is, the quality of training data is guaranteed.

Table 2 Result of candidate generation.

	D1	D2	D3	D4
#cans	5,771	4,258	5,013	482,605
rec	0.9721	0.9535	0.9939	0.9538
	D5	D6	D7	D8
#cans	987,856	38,201,823	61,702,166	46,942,099
rec	0.9780	0.9718	0.9880	0.9970
	D9	D10	D11	D12
#cans	222,686,524	357,365,003	620,073,101	32,161,659
rec	0.9875	0.9770	0.9912	0.9676

4.3 Experiment 1: Candidate Generation

In this experiment, we measure the number of the generated candidates and their recall, which reflects the percentage of the correct candidates over all actual co-references. Because the objective of candidate generator is to retain as many correct candidates as possible, the recall metric is very important. This metric also indicates the maximum recall that *cLink* can obtain for the final resolution result. For computing the recall, we replace the numerator of Eq. (6) by the correct candidates. Table 2 reports the number of the generated candidates *#cans* and the recall *rec*. Although *cLink* uses only the first token for blocking method in the candidate generator, according to this table, the recall cannot reach to 1.0. Meanwhile, the number of candidates is already very large compared to the expected co-references (Table 1), especially on OAEI 2012 dataset (D6 to D12). However, compared to the number of all possible instance pairs between the source and the target repository, more than 99.9% of them are excluded. Obviously, candidate genera-

Table 3 F1 score of *cℒink* when using *cLearn* and other algorithms.

	cLearn	*genetic*	*gain*	*naive*	*none*
D1	0.9365	**0.9375**	0.8204	0.8767	0.8137
D2	0.8679	0.8450	0.8341	**0.8733**	0.8430
D3	**0.8686**	0.7665	0.6841	0.6741	0.6841
D4	**0.6676**	0.6673	0.6651	0.6407	0.5535
D5	0.4725	**0.4727**	0.3065	0.3100	0.2080
D6	**0.8835**	0.8249	0.8760	0.8289	0.7336
D7	**0.9058**	**0.9058**	0.8409	0.8594	0.4508
D8	**0.9645**	0.9635	0.9625	0.9581	0.9506
D9	**0.8781**	**0.8781**	0.8615	0.8609	0.1934
D10	**0.9116**	0.9089	0.8198	0.8105	0.2223
D11	0.9465	**0.9477**	0.9260	0.9325	0.4640
D12	**0.9163**	0.9106	0.8848	0.8908	0.8852
H.Mean	**0.8191**	0.8022	0.7220	0.7236	0.4249

tor considerably reduces the complexity of further components, such as the similarity aggregator and the configuration learner.

4.4 Experiment 2: Compare *cLearn* with Other Algorithms

In order to evaluate the effectiveness of our proposed learning algorithm, we compare the result of *cℒink* when using *cLearn* and when replacing it with other algorithms. We compare *cLearn* with two baseline algorithms: non-optimization (*none*) and naive (*naive*). We also compare *cLearn* with the recent information gain based selection (*gain*), and the state-of-the-art genetic algorithm (*genetic*). *none* works similarly to SLINT+ [4] and SERIMI [3], by accepting all the generated similarities functions (Sect. 3.2) without learning. *naive* selects the K_{top} similarity functions that obtain highest $F1$. *gain* implements the idea of ADL [7], which selects the most discriminative property alignments by measuring the information gain of each property. *genetic* follows the idea of EAGLE [10] and ActiveGenLink [8], which use genetic algorithm to learn the matching configuration. We use binary array representation for the combination of similarity functions. We choose exponential ranking for fitness selection, 0.7 for single point cross-over probability, 0.1 for single point mutation probability, and 50 for the population size. In order to implement other algorithms, we replace the lines from 3 to 19 of Algorithm 1 with the new algorithms. In other words, the mechanism of determining σ_{sim}, δ, and *Agg* remains the same of all algorithms. We are interested in reimplementing other algorithms because it offers more elaborate comparisons. Concretely, it enables using the same input of similarity functions and other settings (e.g., similarity aggregators), which affects the final result. In addition, other algorithms are installed in the systems that are not scalable to large datasets, so that we cannot directly compare *cℒink* with those systems.

We use 5-folds cross validation for this experiment. We choose cross-validation so that all candidates are in turn used for training. Table 3 reports the average F1 scores on each subset of the tested algorithms. According to this table, in overall, *cLearn* consistently outperforms *gain*, *naive*, and *none*. Considers each fold separately, so that there are 60 tests for 12 subsets, the paired t-test over all tests at 0.05 significant level shows that *cLearn* is significantly better than all of *naive*, *gain*, *none*, and *genetic*. Compared to *gain*, *cLearn* consistently outperforms this algorithm for all subsets. Also, although *genetic* is slightly better than *cLearn* on five subsets, the efficiency of *genetic* is much lower than *cLearn*. *genetic* spends averagely 7,231 seconds for learning on one subset of OAEI 2012 *cLearn* only needs 2,977 seconds. The average numbers of configurations that *cLearn* and *genetic* have to check are 126 and 263, respectively. That is, almost 50% configurations are skipped by using *cLearn* compared to *genetic*. This fact supports the efficiency of using our heuristic against the random convergence principle of genetic algorithm.

Table 4 F1 score of *cℒink* and other systems on OAEI 2010.

Training data	System	D1	D2	D3	D4	D5
5%	*cℒink*	**0.911**	**0.824**	**0.777**	0.6414	**0.424**
	Adaboost	0.903	0.794	0.733	0.641	0.375
Variable	*cℒink*	**0.894**	**0.829**	**0.722**		
	ObjectCoref	0.464	0.743	0.708		
Reference systems						
	RiMOM	0.504	0.458	0.629	0.576	0.267
	PARIS	0.649	0.108	0.149	0.502	0.219

4.5 Experiment 3: Compare *cℒink* with Other Systems

We use OAEI 2010 dataset to compare *cℒink* with ObjectCoref [6] and the work in [11], which uses Adaboost to train a classifier. Adaboost uses 5% candidates for training and ObjectCoref uses 20 actual co-references, which is equivalent to 2.3%, 11.6% and 1.2% candidates on D1, D2, and D3, respectively[†]. Therefore, we use the same amount of training data with each other system for *cℒink*, on each respective subset for the comparisons. For each subset, we run *cℒink* 10 times with random selection for training data. After that, we take the average result. In addition, we collect the result of RiMOM and PARIS [17] as two state-of-the-art systems of non-learning based approach for reference.

As shown in Table 4, *cℒink* consistently outperforms other systems. Compared with ObjectCoref, *cℒink* drastically improves the results. Compared with Adaboost, *cℒink* is remarkably better on D3 and D5, which are related to DailyMed, a repository contains the highest number of co-references inside. In overall, learning based systems including *cℒink*, Adaboost, and ObjectCoref are much better than RiMOM and PARIS. This fact confirms the necessity of learning based systems for improving the effectiveness.

For OAEI 2012 datasets, we cannot directly compare with other learning-based systems because none of them supports a large scale dataset like this. The reported result of ADL [7], Knofuss [5], and ActiveGenLink [8] on this dataset is based on a much smaller dataset simplified from

[†]Only the results on D1, D2, and D3 are available for ObjectCoref [26].

Table 5 F1 score of *cLink* and other systems on OAEI 2012.

	cLink	Zhishi.Links	AggrementMaker	SERIMI
D6	0.88	**0.92**	0.69	0.68
D7	0.90	**0.91**	0.74	0.88
D8	0.96	**0.97**	0.88	0.94
D9	0.87	0.88	0.85	**0.91**
D10	0.90	0.87	0.80	**0.91**
D11	0.95	0.93	**0.96**	0.92
D12	0.88	**0.91**	0.85	0.80
H.mean	0.903	**0.912**	0.816	0.853

the original data due to the scalability issue. Using the simple version of this dataset is not difficult to obtain near perfect results, as reported in [4]. However, considering that the learning algorithm is the core of those supervised systems, the results of Experiment 2 partly show the improvement of *cLink* against these systems.

For reference, we report the comparison on OAEI 2012 dataset between *cLink* and three non learning-based systems, including the state-of-the-art Zhishi.Links [18], AgreementMaker [16], and SERIMI [3]. In order to minimize the different of input knowledge, we feed only 5% candidates into the learning algorithm. Table 5 reports this comparison. According to this table, in general, *cLink* clearly outperforms AgreementMaker and SERIMI and is competitive to Zhishi.Links. Note that Zhishi.Links is specially customized for this dataset as this system applies 19 unification rules for matching difficult strings that frequently appear in this dataset (e.g., 'Co' and 'Company', 'Manhattan' and 'NYC'). Therefore, considering the importance of generality *cLink* reveals its strengths against Zhishi.Links.

From above comparisons, *cLink* is competitive or obtains better performance against other systems by using a very limited amount of labeled candidates for training. In the next experiment, we evaluate more detail on the performance of *cLink* with different amount of training data.

4.6 Experiment 4: Size of Training Data

Previous experiment has shown that *cLink* does not need much training data to outperform other systems. However, it is important to know the optimal size of training data from that *cLink* can benefit. We vary the size of training data from 1% to 20% of all candidates and analyze the changing trend of performance. At each ratio, we repeat the random split 10 times for each subset and then measure the average result on the test sets. Figure 2 and Fig. 3 illustrate the results of this experiment. According to these figures, the optimal size of training data is different for each subset. The smallest optimal size is 6% on D8 and the largest is 14% on D6. In overall, consider the harmonic mean, *cLink* only needs 11% candidates for training as after this point, the F1 score does not considerably increase. Although the optimal size of training data is larger than that of Experiment 3, it still can be concluded that *cLink* does not require a large amount of labeled data. This result confirms the practicality of *cLink* and supports the application of supervised learning approach for entity resolution problem.

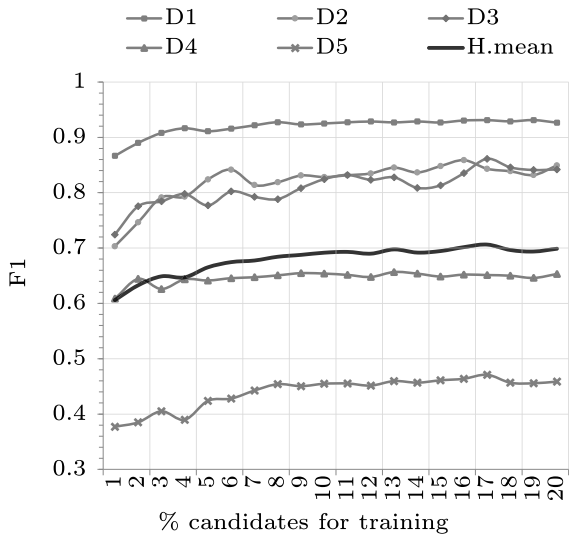

Fig. 2 F1 scores on OAEI 2010.

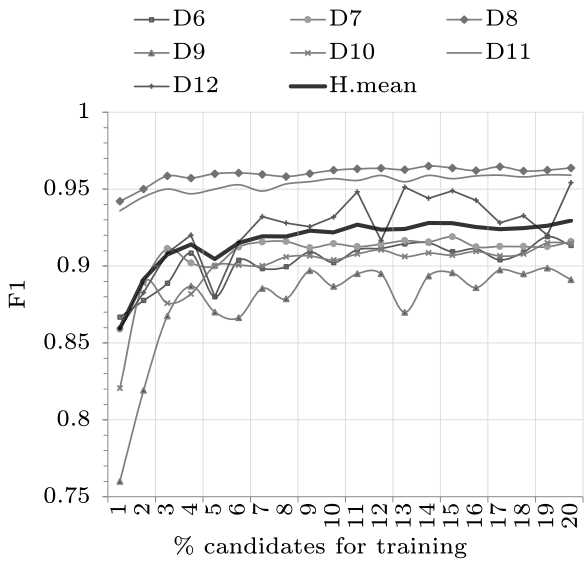

Fig. 3 F1 scores on OAEI 2012.

4.7 Discussion on Similarity Functions and Similarity Aggregators

In order to know which similarity aggregators and similarity measures are most effective and whether they can be universally used or not, we conduct some statistics on the optimal configuration produced by the configuration learning algorithms. We reuse all the results of Experiment 2 and 3, except for Experiment 3, we skip the result of *none* because this algorithm does not perform any similarity function selection.

The selected similarity aggregators are diverse. The most frequently selected similarity aggregators are the two ones with quadratic aggregation ($k = 2$), enabled *restriction*, and *weighting*. Together, these aggregators ap-

pear with 45% out of all cases. Among them, $valid = |U_{F_s im}(x, y)|$ is most frequent on OAEI 2010 (30%) and $valid = 1.0$ is most frequent on OAEI 2012 (28.3%). Similarly, the measures for string similarity are also varied by subset. For Experiment 3, all algorithms produce totally 1,505 similarity functions. Among them, TFIDF Cosine is the most important as its frequency is 41% and is selected for almost every subset. In addition, $rDiff$ is very important for the subsets related to location domain as it is always selected on D6, D9, and D12. The variety of learning results in general and the weak dominance of the most frequent similarity aggregators and similarity measures, show that for particular input, it is difficult for user to define a perfect matching configuration like an automatic system can do.

Another interesting finding is observed for D6. When the size of training data is 80%, only *longitude*, one of two important geographic properties, is selected. While both *longitude* and *latitude* are considered as important for D9, D12, and even in human thinking, the learning algorithm returns a different recommendation. This example, together with the variety of learning results as discussed above, confirm the necessity of automatic learning algorithms.

5. Conclusion and Future Work

In this paper, we presented an effective and efficient supervised instance matching system named *cℒink*. *cℒink* is designed based on the configuration-based instance matching architecture and is enhanced by *cLearn*, a novel heuristic algorithm. *cLearn* can effectively optimize the matching configuration using a small amount of training data. The experimental results show that *cLearn* is significantly better than other algorithms, including the ones that are used by other state-of-the-art systems. Compared to recent systems, *cℒink* also drastically improves the performance.

The candidate generator while delivering high recall, creates many unnecessary candidates. We will investigate learning algorithms for the learning of candidate generator in order to obtain the optimal candidate set. In addition, a study on the stability of the learned configuration would be useful for enabling the utility of transfer learning, from which a configuration can be applied for similar repositories [11]. We also plan to apply active learning to reduce annotation effort for the practical usage of *cℒink*.

References

[1] A. Ferrara, A. Nikolov, and F. Scharffe, "Data linking for the semantic web," International J. Semantic Web and Information System, vol.7, no.3, pp.46–76, 2011.
[2] H. Köpcke and E. Rahm, "Frameworks for entity matching: A comparison," Data & Knowledge Engineering, vol.69, no.2, pp.197–210, 2010.
[3] S. Araujo, D.T. Tran, A. DeVries, J. Hidders, and D. Schwabe, "SERIMI: Class-based disambiguation for effective instance matching over heterogeneous web data," 15th ACM SIGMOD Workshop on the Web and Databases, pp.25–30, 2012.
[4] K. Nguyen, R. Ichise, and B. Le, "Interlinking linked data sources using a domain-independent system," 2nd Joint International Semantic Technology, LNCS, vol.7774, pp.113–128, Springer, 2012.
[5] A. Nikolov, M. d'Aquin, and E. Motta, "Unsupervised learning of link discovery configuration," 9th Extended Semantic Web Conference, LNCS, vol.7295, pp.119–133, Springer, 2012.
[6] W. Hu, J. Chen, and Y. Qu, "A self-training approach for resolving object coreference on the semantic web," 20th International Conference on World Wide Web, pp.87–96, 2011.
[7] W. Hu, R. Yang, and Y. Qu, "Automatically generating data linkages using class-based discriminative properties," Data & Knowledge Engineering, vol.91, pp.34–51, 2014.
[8] R. Isele and C. Bizer, "Active learning of expressive linkage rules using genetic programming," Web Semantics: Science, Services and Agents on the World Wide Web, vol.23, pp.2–15, 2013.
[9] A.C.N. Ngomo, J. Lehmann, S. Auer, and K. Höffner, "RAVEN - Active learning of link specifications," 6th International Semantic Web Conference Workshop on Ontology Matching, pp.25–36, 2011.
[10] A.C.N. Ngomo and K. Lyko, "EAGLE: Efficient active learning of link specifications using genetic programming," 9th Extended Semantic Web Conference, LNCS, vol.7295, pp.149–163, Springer, 2012.
[11] S. Rong, X. Niu, W.E. Xiang, H. Wang, Q. Yang, and Y. Yu, "A machine learning approach for instance matching based on similarity metrics," 11th International Semantic Web Conference, LNCS, vol.7649, pp.460–475, Springer, 2012.
[12] K. Nguyen and R. Ichise, "Heuristic-based configuration learning for linked data instance matching," 5th Joint International Semantic Technology Conference, 2015.
[13] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, "Discovering and maintaining links on the web of data," 8th International Semantic Web Conference, LNCS, vol.5823, pp.650–665, Springer, 2009.
[14] A.C.N. Ngomo and S. Auer, "LIMES: A time-efficient approach for large-scale link discovery on the web of data," 22nd International Joint Conference on Artificial Intelligence, pp.2312–2317, 2011.
[15] K. Nguyen and R. Ichise, "ScSLINT: Time and memory efficient interlinking framework for linked data," 14th International Semantic Web Conference Posters and Demonstrations Track, 2015.
[16] I.F. Cruz, F.P. Antonelli, and C. Stroe, "AgreementMaker: Efficient matching for large real-world schemas and ontologies," VLDB Endowment, vol.2, pp.1586–1589, 2009.
[17] J. Li, J. Tang, Y. Li, and Q. Luo, "RiMOM: A dynamic multistrategy ontology alignment framework," IEEE Trans. Knowledge and Data Engineering, vol.21, no.8, pp.1218–1232, 2009.
[18] X. Niu, S. Rong, Y. Zhang, and H. Wang, "Zhishi.links results for OAEI 2011," 6th ISWC Workshop on Ontology Matching, pp.220–227, 2011.
[19] F.M. Suchanek, S. Abiteboul, and P. Senellart, "PARIS: Probabilistic alignment of relations, instances, and schema," VLDB Endowment, vol.5, no.3, pp.157–168, 2011.
[20] M. Niepert, C. Meilicke, and H. Stuckenschmidt, "A probabilistic-logical framework for ontology matching," 24th Conference on Artificial Intelligence, pp.1413–1418, 2010.
[21] G. Demartini, D.E. Difallah, and P. Cudré-Mauroux, "Large-scale linked data integration using probabilistic reasoning and crowdsourcing," The VLDB Journal, vol.22, no.5, pp.665–687, 2013.
[22] D. Song and J. Heflin, "Automatically generating data linkages using a domain-independent candidate selection approach," 10th International Semantic Web Conference, LNCS, vol.7031, pp.649–664, Springer, 2011.
[23] M. Bilenko, B. Kamath, and R.J. Mooney, "Adaptive blocking: Learning to scale up record linkage," 6th International Conference on Data Mining, pp.87–96, 2006.
[24] K. Nguyen and R. Ichise, "A heuristic approach for configuration learning of supervised instance matching," 14th International Semantic Web Conference Posters and Demonstrations Track, 2015.
[25] D. Gale and L.S. Shapley, "College admissions and the stability of marriage," American Mathematical Monthly, vol.96, no.1, pp.9–15,

1962.

[26] W. Hu, J. Chen, G. Cheng, and Y. Qu, "Objectcoref & falcon-ao: results for oaei 2010," 5th ISWC Workshop on Ontology Matching, pp.158–165, 2010.

Khai Nguyen received his MSc. degree in computer science from the University of Science, Ho Chi Minh City, Vietnam, in 2012. He is currently a Ph.D. candidate at the Graduate University for Advanced Studies in Japan. His research interests include linked data, semantic web, and data mining.

Ryutaro Ichise received his Ph.D. degree in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 2000. From 2001 to 2002, he was a visiting scholar at Stanford University. He is currently an associate professor in the Principles of Informatics Research Division of the National Institute of Informatics, Japan. His research interests include machine learning, semantic web, and data mining.