Interlinking Linked Data Sources Using a Domain-Independent System

Khai Nguyen[1], Ryutaro Ichise[2], and Bac Le[1]

[1] University of Science, Ho Chi Minh, Vietnam
{nhkhai,lhbac}@fit.hcmus.edu.vn
[2] National Institute of Informatics, Tokyo, Japan
ichise@nii.ac.jp

Abstract. Linked data interlinking is the discovery of every *owl:sameAs* links between given data sources. An *owl:sameAs* link declares the homogeneous relation between two instances that co-refer to the same real-world object. Traditional methods compare two instances by predefined pairs of RDF predicates, and therefore they rely on the domain of the data. Recently, researchers have attempted to achieve the domain-independent goal by automatically building the linkage rules. However they still require the human curation for the labeled data as the input for learning process. In this paper, we present SLINT+, an interlinking system that is training-free and domain-independent. SLINT+ finds the important predicates of each data sources and combines them to form predicate alignments. The most useful alignments are then selected in the consideration of their confidence. Finally, SLINT+ uses selected predicate alignments as the guide for generating candidate and matching instances. Experimental results show that our system is very efficient when interlinking data sources in 119 different domains. The very considerable improvements on both precision and recall against recent systems are also reported.

Keywords: linked data, interlinking, domain-independent, instance matching.

1 Introduction

The linked data is an unsubstitutable component in the generation of semantic web. It provides a mechanism by which the resources are interconnected by structured links. These links not only help the representation of information becomes clearer, but also makes the exploitation of information more efficient. Since then, linked data gets a lot of interest from many of organizations and researchers. Many linked data sources are developed and many support tools are introduced.

Given two linked data sources, data interlinking discovers every *owl:sameAs* links between these sources. An *owl:sameAs* link describes the homogeneity the instances that refer to the same object in the real world. Data interlinking is used in two typical processes: data construction and integration. When publishing linked data, an essential task is to declare the links between the instances to ensure the "linked" property of the data. Among these links, finding the *owl:sameAs* is prominently important and challenging [2]. In data integration, because many data sources are independently developed, the consideration of the homogeneous instances is important since it ensures the integrity and consistency of data.

We attempt to develop an interlinking system that is independent with the domain of data. This goal is increasingly interested because linked data is spreading over many areas. Since a linked data instance is represented by a set of RDF triples (subject, predicate, and object), the schema of the data sources is equivalent with the list of used RDF predicates. Frequently, different data sources may use different RDF predicates to describe the same property of the instances. Because the properties are commonly regulated by the domain of the data, difference in domain has the same meaning with difference in schema.

Traditional approaches compare two instances by matching the RDF objects that declared by the corresponding predicates and these alignments are manually mapped by the human [3,8,10]. This approach is inapplicable when the users do not have enough knowledge about the data and manually generating predicate alignments may ignore the hidden useful ones. Recently, researchers have attempted to learn the linkage rules [5,6] for not relying on the domain. However, most approaches require labeled data, which is still involved with human curation.

In this paper, we introduce SLINT+, a training-free system with a domain-independent approach. Our system firstly collects the important predicates of each data sources using the covering and discriminative abilities of predicates. Then, it combines these predicates and selects the most appropriate alignments by considering their confidence. The selected alignments are the guide for comparing instances in the final step, instance matching. While instance matching produces all the *owl:sameAs* links between data sources, the preceding step is the candidate generation, which extracts the potentially homogeneous instances. The basic idea of candidate generation step is quickly comparing the collective information of instances using collected predicates. While the key requirements of an interlinking system are mainly the recall and precision, those of candidate generation are the pair completeness and reduction ratio. These criteria are used to evaluate our system in the experiment. For testing the domain-independent ability, we use 1,6 million *owl:sameAs* links connecting DBpedia[1] and Freebase[2] with 119 different domains. The experimental results are the evidences for the efficiency of SLINT+. Besides, we compare SLINT+ with the previous state-of-the-art systems and report the considerable improvements.

The paper is structured as follows. In the next section, we review some representative linked data interlinking approaches and systems. Section 3 is the detail description about SLINT+. Section 4 reports the experiments as well as the re-

[1] http://dbpedia.org/

[2] http://www.freebase.com/

sults and analyses. Section 5 closes the paper with the conclusions and future directions.

2 Related works

One of the first linked data interlinking system is Silk [10]. Silk is a link discovery framework that provides a declarative language for user to define the type of link to discover. The main use of Silk is to find *owl:sameAs* links. Using Silk, users need to declare the pairs of predicates that they want to compare when matching the instances. Besides, the matching threshold is manually configured. AgreementMaker [3] is known as a system that focuses on matching both ontologies and instances. In data interlinking, AgreementMaker use a three steps matching process, including candidate generation, disambiguation, and matching. Candidates are collected by picking the instances that share the same label with others. The disambiguation step divides the candidates into smaller subsets and the matching step verify every pair of instances in each subset to produce the final result. Zhishi.Links [8] is one of the current state-of-the-art systems. This system improves the matching efficiency by using weighting schemes (e.g. TF-IDF, BM25) for RDF objects when generating candidate, as an adoption of pre-matching phase of Silk. However, this system is still depending on the domain of data.

Recently, many domain-independent approaches have been proposed [1,5,6,9]. SERIMI [1] selects the useful predicates and their alignments by considering the entropy and the similarity of RDF objects. SERIMI is the second best system at the OAEI Instance Matching 2011 [4]. Isele and Bizer presented a linkage rule generation algorithm using genetic programming [5]. Nguyen et al. focused on instance matching using learning approach [6]. They build a binary classifier to detect the matched and non-matched pairs of instances. Song and Heffin proposed a domain-independent candidate generation method [9]. They design an unsupervised learning schema to find the most frequent and discriminative predicates. The set of predicates in each data sources are used as the key for candidate generation.

In general, most proposed domain-independent approaches use the labeled data as the replacement for user knowledge about the schema or the domain of the data. Our innovation is developing SLINT+, an interlinking system that does not need any information about both the domain and the matching state of a portion of data sources. SLINT+ is an extension of SLINT [7]. Comparing with SLINT, SLINT+ is similar in the architecture of the system, and different in the use of techniques in some steps. The major improvement of SLINT+ against SLINT is the reduction of many manual thresholds. In the next section we will describe the technical detail of SLINT+.

3 Domain-Independent linked data interlinking system

In this section, we describe the SLINT+ system. The process of interlinking two data sources D_S and D_T is summarized in Fig. 1. There are four ordered steps in

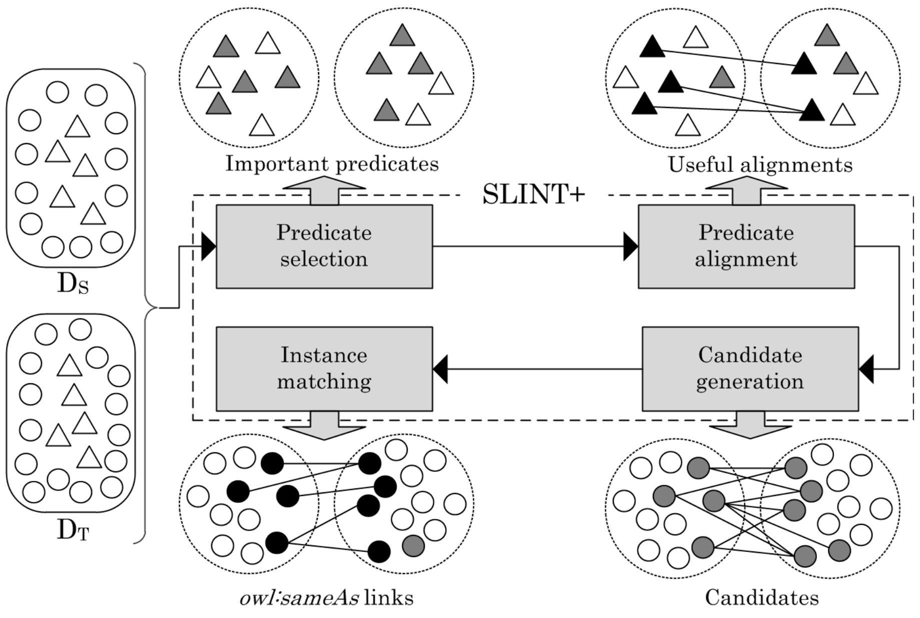

Fig. 1. The interlinking process of SLINT+

this process: predicate selection, predicate alignment, candidate generation, and instance matching. The predicate selection step aims at finding the important predicates, which are illustrated as the dark triangles in the figure. This kind of predicates in both source data D_S and target data D_T are then combined to construct the raw predicate alignments in the next step. We collect the useful alignments by comparing their confidence level with an automatically selected threshold. While the first two steps resolve the domain-independent objective, the last two steps perform the interlinking using the output of the previous ones. The candidate generation step is installed to find the candidates, which are the potentially homogeneous instances and are expected to be very small in quantity compared with all possible pairs of instances. In Fig. 1, the candidates are visualized as the connected pairs of small dark circles in the output of the third step. SLINT+ only conducts the comparison for these candidates to produce the final *owl:sameAs* links. In the following sections, we give the detail of each step.

3.1 Predicate selection

The predicate selection step is designed to find the important predicates. We assume that important predicates are the ones used to declare the common properties and the distinct information of the object. Therefore, we implement two criteria of an important predicate: the coverage and the discrimination. The coverage of a predicate expresses its frequency while the discrimination represents the variation of the RDF objects described by this predicate. In Eq.1

and Eq.2, we define the coverage $cov(p, D)$ and discrimination $dis(p, D)$ of a predicate p in the data source D.

$$cov(p, D) = \frac{|\{x | \exists < s, p, o > \in x, x \in D\}|}{|D|}. \tag{1}$$

$$\begin{aligned} & dis(p, D) = HMean(V(p, D), H(p, D)) \\ & Where \\ & V(p,D) = \frac{|\{o | \exists x \in D, < s, p, o > \in x\}|}{|\{< s, p, o > | \exists x \in D, < s, p, o > \in x\}|} \\ & H(p,D) = \sum_{w_i \in O} \frac{f(w_i)}{\sum_{w_j \in O} f(w_j)} \log \frac{f(w_i)}{\sum_{w_j \in O} f(w_j)} \\ & O = \{o | \exists x \in D, < s, p, o > \in x\}. \end{aligned} \tag{2}$$

In these equations, x represents an instance, which is a set of RDF triple $< s, p, o >$ (subject, predicate, object). Function f returns the frequency of input RDF object in the interested data source. Using the coverage and discrimination, a predicate is considered to be important if it satisfies the condition of Eq.3.

$$\begin{cases} cov(p, D) & \geq \alpha \\ dis(p, D) & \geq \beta \\ HMean(cov(p, D), dis(p, D)) & \geq \gamma. \end{cases} \tag{3}$$

The coverage of a predicate is the percent of the instances sharing it. This metric is the first requirement of an important predicate because we aim at finding the predicates that are used to describe the common properties of the instances. The number of important predicates is considerably small when being compared with all predicates. For example, in 24,110 instances of *school* domain in DBpedia, there are 2,771 distinct predicates but only 1% of them, 28 predicates, has the coverage that is over 0.5. On another hand, the discrimination is the harmonic mean of the diversity and the entropy of RDF objects. The function H yields the entropy of distinct RDF objects, while V returns the percent of them over total triples. V is a good option to estimate the variation of RDF objects, and H is used to reveal the difference of the predicates that share the same number of distinct RDF objects but differ in the distribution of each value. Clearly, a predicate that describes the ambiguous information of instances should not be important. Therefore, we install the discrimination as the second criterion of an important predicate.

The α and β are automatically configured using the average value of the coverage and discrimination of all predicates, respectively. In another word, α and β are set to α_{mean} and β_{mean}, respectively. The γ is the main requirement when being used to select the predicates having high harmonic mean of coverage and discrimination. Usually, α_{mean} and β_{mean} are quite small because the percent of important predicates is not large. Therefore, γ should have a value larger than α_{mean} and β_{mean} to correctly select the important predicates.

Table 1. Representative extraction function R

Type	Return values of R
string	String tokens
URI	String tokens (separator '/', domain is omitted)
decimal	Rounded values with 2 decimal numbers
integer	Original values
date	Original values

The idea of Eq.3 and function V are inherited from the work of Song and Heffin [9]. We also implemented V as the discrimination function in SLINT [7]. In SLINT+, we extend the discrimination as the combination of V and entropy function H. Besides, we use the α_{mean} and β_{mean} instead of manually configuring them as in [7,9].

For each input data source, SLINT+ collects the important predicates of each source and forwards them into the next step, predicate alignment.

3.2 Predicate alignment

The aim of this step is to collect the useful alignments of important predicates. A predicate alignment is a pair of two predicates from source data and target data. A useful alignment is expected to be the predicates that describe the same information of the existing instances in each data source. We construct predicate alignments by combining the predicates having the same data type, and after that, we select the ones having high confidence for the result. We categorize RDF predicates into five different types: *string*, *URI*, *decimal*, *integer*, and *date*. The type of a predicate is determined using the major type of RDF objects, which are accompanied by this predicate. For example, predicate p is used to declare the values that can be *string* and *URI* if the frequency of the manner is higher than the latter, then the data type assigned for p will be *string*.

We estimate the confidence of an alignment using the similarity of the representatives of RDF objects. The representatives are the pre-processed values of them. Denoting R as the representative extraction function, the return values of R is given in Table 1. The confidence $conf(p_S, p_T)$ of the alignment between two predicates p_S and p_T is then computed using Eq.4

$$conf(p_S, p_T) = \frac{2 \times |R(O_S) \cap R(O_T)|}{|R(O_S)| + |R(O_T)|}, O_k = \{o | \exists x \in D_k, < s, p_k, o > \in x\}. \quad (4)$$

The idea of Eq.4 starts from the assumption in that the corresponding predicates are used to describe the same properties of the instances. Therefore, the higher value of confidence is, the more useful an alignment is. We use thresholding approach for choosing the useful alignments with the threshold δ is set to the average confidence δ_{mean} of all alignments. Because there are many meaningless alignments when combining the predicates of two data sources, we ignore the low confidences. Therefore, we take the average value of the confidences larger than ϵ, a small value.

Predicate alignment step is the key difference in the interlinking model of SLINT+ against previous systems. Other domain-independent approaches [1,5,6,9] directly generate linkage rules and do not find the corresponding predicates. We attempt a new approach, in which the linkage rules are implicitly defined, and they are the useful alignments.

The two first steps of SLINT+ are the solution for the domain-independent ability. In the next steps, we use their results to perform the candidate generation and instance matching.

3.3 Candidate generation

When interlinking two data sources D_S and D_T, each instance in D_S should be compared with each instance in D_T. However, conducting all the comparisons is impractical. Candidate generation step aims at collecting all pairs of instances that have a high possibility to be homogeneous. These kind of pair is called *candidate*. Since the basic idea of every set-matching problem is exhausting similitude, candidate generation should be designed not to perform explicit comparisons. There are three consecutive parts in this step: indexing, accumulating, and candidate selection. The first part indexes every instance in each data source with the representative value of RDF objects. The second part builds the weighted co-occurrence matrix to store the co-occurrence value of all pairs of instances. The last part selects the candidate by considering the high elements in the matrix. We put the summary of our candidate generation method in Algorithm 1. In this algorithm, Pr_s and Pr_t represent the predicates that appear in the useful alignments, where Pr_k belongs to D_k. H, M, C, Rp represent the index table, weighted co-occurrence matrix, candidates set, and representative extraction function, respectively. ζ is a factor that will be combined with max to produce the threshold for candidate selection, as written in line 22.

In Algorithm 1, lines 4-11 describe the indexing process. The result of this part is the inverted-index table H. An entry of H is formed by two elements: the key $r.Label$, and the triples $< D, x, r.Value \times sumConf >$. In a triple, the first two elements D and x indicate the identifier of instance $x \in D$ (e.g. index of x in D), which contains $r.Label$ in the representative set of its RDF objects. The last element is the weight of $r.Label$. If an entry of H contains n triples, there will be n instances sharing the same key of this entry. In the design of function Rp, $r.Label$ is extracted as the same manner with function R (Table 1), and $r.Value$ is regulated by the data type of interested predicate. For *string* or *URI*, $r.Value$ is set to the TF-IDF score of the token. For *decimal*, *integer*, or *date*, $r.Value$ is fixed to 1.0.

Lines 12-15 are the building process of co-occurrence matrix M. An element of this matrix is the co-occurrence value of two instances from source data and target data. We use the confidence of useful predicate alignments as the weight for each accumulated value. In addition with the inverted-indexing, the accu-

Algorithm 1: Generating candidates set

```
   Input: $D_S$, $D_T$, $Pr_S$, $Pr_T$, $\zeta$
   Output: Candidate set $C$
1  $H \leftarrow \emptyset$
2  $M[|D_S|, |D_T|] \leftarrow \{0\}$
3  $C \leftarrow \emptyset$
4  foreach $< D, P > \in \{< D_S, Pr_S >, < D_T, Pr_T >\}$ do
5      foreach $x \in D$ do
6          foreach $p_i \in P$ do
7              $sumConf \leftarrow \sum_{p_j \in \{Pr_S, Pr_T\} \setminus P} conf(p_i, p_j)$
8              foreach $r \in Rp(O), O = \{o| < s, p_i, o > \in x\}$ do
9                  if not $H.ContainsKey(r.Label)$ then
10                     $H$.AddKey($r.Label$)
11                 $H$.AddValue($r.Label$, $< D, x, r.Value \times sumConf >$)
12 foreach $key \in H.AllKeys()$ do
13     foreach $< x_S, v_S > \in H.GetIndices(key, D_S)$ do
14         foreach $< x_T, v_T > \in H.GetIndices(key, D_T)$ do
15             $M[x_S, x_T] \leftarrow M[x_S, x_T] + v_S \times v_T$
16 $\lambda = \text{Mean}(M)$
17 foreach $x_S \in D_S$ do
18     foreach $x_T \in D_T$ do
19         $max_S \leftarrow \text{Max}(M[x_S, x_j]), \forall x_j \in D_T$
20         $max_T \leftarrow \text{Max}(M[x_i, x_T]), \forall x_i \in D_S$
21         $max \leftarrow \text{HMean}(max_S, max_T)$
22         if $M[x_S, x_T] \geq \lambda$ and $M[x_S, x_T] \geq \zeta \times max$ then
23             $C \leftarrow C \cup < x_S, x_T >$
24 return $C$
```

mulating process improves the speed of candidate generation because each data source and the index table need to be traversed one time.

Lines 16-23 are the candidate selection process. We apply an adaptive filtering technique by using the data driven thresholds. λ is installed to warranty there is no assumption about the surjection. This threshold is assigned to the average co-occurrence value of all pairs of instances. This value is usually small because the number of homogeneous pairs is frequently very much lower than that of the non-homogeneous ones. ζ is manually configured and timed with an automatically selected value max to produce the dynamic threshold $\zeta \times max$. This threshold is the main requirement of a homogeneous pair of instances.

Comparing with previous systems, our method is distinct for the weighted co-occurrence matrix and adaptive filtering in candidate selection. While proposed methods compare one or a few pairs of RDF objects, we aggregate multiple similarities from many pairs of RDF objects for improving the quality of "rough" similarity of instances. For candidate selection, SERIMI [1] and Song et al. use traditional *thresholding* [9]. SLINT+ also use this approach as the availability of λ. However, the impact of λ is very small because the value assigned for it is quite low and the key idea of the selection method is adaptive filtering. Silk [10] and Zhishi.Links [8] selects k best correlated candidates for each instance. This approach has advantage in handling the number of candidates, but the fixed value for k seems not reasonable because the ambiguous level are not the same for each instance. While some instances should have small value of k, the others must have larger candidates for still including the homogeneous ones. In the next step, SLINT+ verifies all selected candidates for producing the final *owl:sameAs* links.

3.4 Instance matching

We estimate the correlation of each candidate and the ones satisfying the process of adaptive filtering will be considered to be homogeneous. The correlation of a candidate, or of two instances, is computed using the similarity of RDF objects of interested instances. These objects are described using useful predicate alignments and the final correlation of two instances is the weighted average of these similarities. The correlation $corr(x_S, x_T)$ of two instances $x_S \in D_S$ and $x_T \in D_T$ is defined in Eq.5.

$$\begin{aligned}
corr(x_S, x_T) =& \frac{1}{W} \sum_{<p_S,p_T>\in A} conf(p_S,p_T) \times sim(R(O_S), R(O_T)),\\
&Where\\
&\quad O_k = \{o | \exists x \in D_k, < s, p_k, o > \in x\}\\
&\quad W = \sum_{<p_S,p_T>\in A} conf(p_S,p_T).
\end{aligned} \tag{5}$$

In this equation, A is the set of useful predicate alignments; R is the representative extraction function, which is the same as that in Table 1; $conf(p_S,p_T)$ is the confidence of interested predicate alignment $< p_S, p_T >$ (Eq.3). The *sim* function returns the similarity of two set of representatives, and is regulated by the data type of the predicates. For *decimal* and *integer*, we use the variance of representative values. For *date*, we use exact matching and return the value 1 or 0 when the values are totally equal or not, respectively. For *string* and *URI*, we take the cosine similarity with TF-IDF weighting, as given in Eq.6. While cosine is widely used to estimate the similarity of two sets, the TF-IDF weighting very helpful for many disambiguation techniques.

$$sim(Q_s, Q_t) = \frac{\sum_{q\in Q_s\cap Q_t} TFIDF(q,Q_s)TFIDF(q,Q_t)}{\sqrt{\sum_{q\in Q_s} TFIDF^2(q,Q_s) \times \sum_{q\in Q_t} TFIDF^2(q,Q_t)}}. \tag{6}$$

After computing correlation value for every candidate, we apply adaptive filtering to collect homogeneous pairs. An *owl:sameAs* link is created when two instances have the score larger than a threshold, which is dynamically adjusted in accordance with the interested instances. Denoting C as the output candidates of Algorithm 1, we define the set of homogeneous instances I as in Eq.7.

$$I = \{< x_S, x_T > | corr(x_S, x_T) \geq \eta \wedge \\ \frac{corr(x_S, x_T)}{max_{\forall < x_m, x_n > \in C, x_m \equiv x_S \vee x_n \equiv x_T} corr(x_m, x_n)} \geq \theta\}. \quad (7)$$

We assume that an *owl:sameAs* link connects two instances that have the highest correlation if compared this value with the correlation of other candidate, in which each instance appears. However, θ threshold is necessary to be installed to select the pairs of instance that are co-homogeneous, because we do not assume that there is no duplication in each data sources. In SLINT+, θ is set to a quite large value. We use the average correlation η_{mean} of all candidates for configuring η. Since there are frequently many candidates that are not homogeneous, η_{mean} is usually small. Like λ in Algorithm 1, η ensures there is no assumption about the surjection of given data sources and should be assigned with a low value.

Compared with previous systems, the instance matching step of SLINT+ is distinct in the use of weighted average and adaptive filtering. We use adaptive thresholds for each pair of data sources and the confidence of useful alignments for weighting. Silk [10] also provides a weighted average combination method for similarities, however these weights must be configured manually instead of observing from the data. Zhishi.Links [8] and AgreementMaker[3] select the best correlated candidates, while Silk [10] and SERIMI [1] use traditional threshold-based approach. In the next section, we will report our experiments and the results.

4 Experiments

4.1 Evaluation metrics

We evaluate the efficiency of the interlinking process of SLINT+ at two main metrics: recall and precision. The recall represents the ability that retains the true *owl:sameAs* links, while the precision indicates the disambiguation ability because it expresses the percent of true elements in all discovered links. In addition to precision and recall, we also report the $F1$ score, the harmonic mean of them. Eq. 8 and 9 are the computation of recall RC and precision PR, respectively.

$$RC = \frac{Number\ of\ correctly\ discovered\ links}{Number\ of\ actual\ links}. \quad (8)$$

$$PR = \frac{Number\ of\ correctly\ discovered\ links}{Number\ of\ all\ discovered\ links}. \quad (9)$$

For evaluating the candidate generation, we also use two main metrics, the pair completeness and reduction ratio. The pair completeness expresses the percent of correct generated candidates and the reduction ratio shows the compactness of candidate set. The aim of candidate generation is to maximize the pair completeness while reserving a very high reduction ratio. Eq. 10 and 11 show the formula of pair completeness PC and reduction ratio RR, respectively. Some studies also use the $F1$ score for PC and RR [9], however we think that it is not equivalent to combine these metrics because RR is frequently very high in compared with PC in most cases.

$$PC = \frac{Number\ of\ correct\ candidates}{Number\ of\ actual\ \ links}. \tag{10}$$

$$RR\ =\ 1\ -\ \frac{Number\ of\ candidates}{Number\ of\ all\ instance\ pairs}. \tag{11}$$

We also report the execution times of SLINT+ in the division of three parts, predication selection and predicate alignment, candidate generation, and instance matching. Every experiment is conducted on a desktop machine with 2.66Ghz of CPU, 8GB of memory, 32-bit Windows 7 operating system. In addition, we use C# language for developing SLINT+.

4.2 Experiment setup and datasets

Since SLINT+ contains few manual thresholds, we use the same value for each threshold on every dataset to evaluate the domain-independent ability. We set the value 0.5, 0.1, 0.5, and 0.95 for γ (Eq.3), ϵ (Section 3.2), ζ (Algorithm 1), and θ (Eq. 7), respectively.

The aim of experiment is to evaluate the efficiency of SLINT+, especially the domain-independent goal. Besides, a comparison with existing systems is also necessary. Therefore, we use two datasets DF and OAEI2011. The first dataset, DF is selected from a ground-truth provided by DBpedia[3]. This set contains about 1,6 million *owl:sameAs* links between DBpedia and Freebase, and 141 domains of data. DBpedia and Freebase are the most well-known and are very large data sources in the linked data community. DBpedia and Freebase contains many domains such as people, species, drugs, rivers, songs, films, settlements,... For constructing the DF, we pick up the domains that have the number of *owl:sameAs* links is at most 40,000. After this selection, we have 891,207 links with 119 domains. We divide these links into 119 subsets and each subset is respective to a distinct domain. The smallest subset contains 1,008 links in *architect* domain and the largest subset contains 37,120 links about *plant*. We do not use the domains that contain more than 40,000 instances because the computation of candidate generation requires a large amount of memory, which is the limitation of our testing environment. However, the number 119 is still convinced for verifying the domain-independent ability.

The second dataset, OAEI2011 is the data that was selected for the most recent OAEI 2011 Instance Matching Campaign [4]. In this campaign, participants are asked to build the *owl:sameAs* links between NYTimes[4] to DBpedia, Freebase, and Geonames[5]. NYTimes is a high quality data source while DBpe-

[3] http://downloads.dbpedia.org/3.8/links/freebase_links.nt.bz2
[4] http://data.nytimes.com/
[5] http://www.geonames.org/

Table 2. Domains and numbers of owl:sameAs links in OAEI2011 dataset

ID	Source	Target	Domain	Links
D1	NYTimes	DBpedia	Locations	1920
D2	NYTimes	DBpedia	Organizations	1949
D3	NYTimes	DBpedia	People	4977
D4	NYTimes	Freebase	Locations	1920
D5	NYTimes	Freebase	Organizations	3044
D6	NYTimes	Freebase	People	4979
D7	NYTimes	Geonames	Locations	1789

Table 3. Number of predicates and predicate alignments in DF dataset

	Pr_S	Pr_T	P_S	P_T	A	K
Min	228	9	7	4	12	6
Max	2711	1388	30	21	158	40
Average	468	315	14	8	45	16

dia, Freebase, and Geonames are very large ones. Geonames is the data sources focusing on geography domain and currently contains over 8,0 million geographic names. The OAEI2011 dataset has 7 subsets and 3 domains, which are *location*, *organization*, and *people*. Denoting the 7 subsets as D1 to D7, the overview of this dataset is given in Table 2.

In the next sections, we report the results and the discussions for each step of SLINT+.

4.3 Results and analyses

4.3.1 Discussion on predicate selection and predicate alignment

Table 3 gives the number of all predicates Pr in each data source, important predicates P, all predicate alignments A and the useful alignments K in DF dataset. S and T stand for DBpedia and Freebase, respectively. According to this table, the numbers of all predicates are very high when being compared with the numbers of important predicates at the threshold $\gamma = 0.5$, which means that the requirements for covering and discriminative abilities are around 0.5. If we consider the percent of important predicates with this threshold, the maximum values are 25.00% and 17.24% in DBpedia and Freebase, respectively; and the averages are only 4.95% and 3.70% in DBpedia and Freebase, respectively. The low ratio of important predicates indicates the high heterogeneity of the predicates in the schema of data sources. The predicate selection step is therefore necessary to detect the useful predicates. As our observation, the selected predicates are frequently the predicates describing the name or the label of the instances, and some important predicates that rely on the domain of the data. For example, in *city* domain, beside the predicate declaring the name, the useful predicates are also about the longitude and latitude of the city. This example also indicates the *string* data type is not the only important one although this assumption is right in many cases.

Table 4. Results of candidate generation on DF dataset

	NC	PC	RR
Min	1238	0.9652	0.9969
Max	389034	1.0000	0.9999
Weighted average	87089	0.9890	0.9996

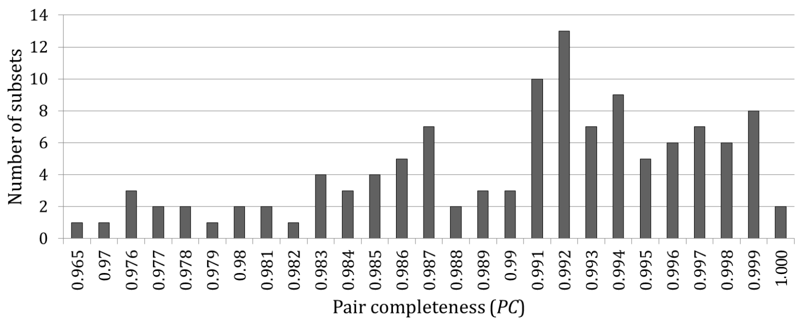

Fig. 2. Histogram of subsets on their pair completeness

For predicate alignment, since the number of important predicates is reduced in the first step, the number of all alignments is not large when the maximum value is only 158. In detail, the selected alignments, which are higher than the average confidence, occupy 42.70% of all alignments at average, 91.30% at the maximum, and 12.50% at the minimum. As our observation, the useful predicate alignments are very appropriate when the predicates describing the similar property are always combined. For example, in *philosopher* domain, the selected alignments contains 3 interesting ones: the combination of 3 different DBpedia predicates describing full name, surname, and given name with 1 Freebase predicate describing the name of the person. It is not easy for a user to point out every useful alignment, especially in the case that useful alignments can reach the number of 16, as the average value in Table 3.

There is currently no dataset for clearly evaluating the efficiency of predicate selection and predicate alignment. However, the high results of the candidate generation and the whole interlinking process are the very clear evidences for the efficiency of these tasks.

4.3.2 Candidate generation

In this section, we report the results of candidate generation on DF dataset. Table 4 summarizes the candidate generation results. In this table, NC is the number of candidates, and the weight of weighted average values are the numbers of *owl:sameAs* links in each subset. The pair completeness is very high when candidate generation step reserves over 96.52% of correct candidates. Fig. 2 is the histogram of subsets when their PC is rounded into the nearest value in the horizontal axis. According to this figure, most PCs are distributed in the

Table 5. Results of data interlinking on DF dataset

	RC	PR	$F1$
Min	0.9002	0.8759	0.8985
Max	0.9973	1.0000	0.9986
Weighted average	0.9763	0.9645	0.9702

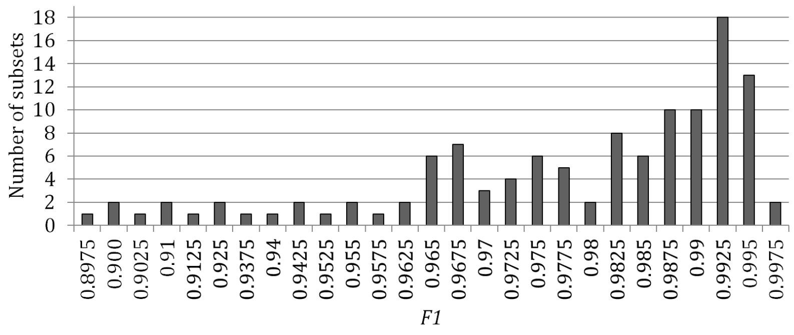

Fig. 3. Histogram of subsets on their F1 score

range above 0.991 and reach the highest number at 0.992. The subsets having the PC that is higher than 0.991 are 73 and cover 61.34% of all the subsets in DF. The reduction ratio RR in Table 4 is very impressive when always higher than 0.9969 on every subset. The adaptive filtering technique is also verified through the high result of pair completeness and reduction ratio. In general, the candidate generation step has successfully performed its role on over 119 subsets.

4.3.3 The whole interlinking process

For the whole interlinking process, we report the result of recall RC, precision PR, and $F1$ score. As summarized in Table 5, both recall and precision are very considerable when the weighted average value for $F1$ is very high at 0.9702. Fig. 3 show the histogram of subsets when considering their $F1$ scores. There are 82.35% of subsets having the $F1$ higher than 0.965, the middle value of horizontal axis. These subsets cover 79.40% of all *owl:sameAs* links in the DF dataset. The high precision and recall not only express the efficiency of interlinking, but also reveal that the predicate selection and predicate alignment produced useful predicates and appropriate alignments.

Fig. 4 shows the runtime of predicate selection and predicate alignment, candidate generation, and instance matching in accordance with the size of the subsets. There are a few points that the runtime does not consistently increase because beside the size of data, the number of predicates also affects the speed of each step. On large subsets, the time-variation of the each step is clearer and we can see that the major time of interlinking process is the candidate generation. If we consider the weighted average runtime, this step occupies 54.45% of

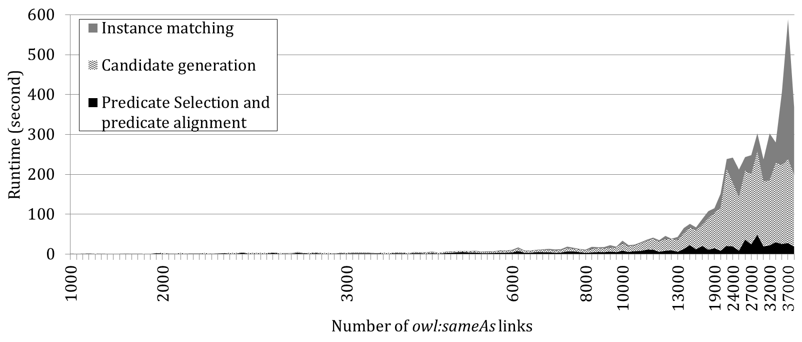

Fig. 4. Execution time of interlinking steps

Table 6. Comparison with previous interlinking systems on OAEI2011 dataset.

Dataset	SLINT+			Agree.Maker			SERIMI			Zhishi.Links		
	PR	RC	$F1$	PR	RC	$F1$	PR	RC	$F1$	PR	RC	$F1$
D1	**0.96**	**0.97**	**0.97**	0.79	0.61	0.69	0.69	0.67	0.68	0.92	0.91	0.92
D2	**0.97**	**0.95**	**0.96**	0.84	0.67	0.74	0.89	0.87	0.88	0.90	0.93	0.91
D3	**0.99**	**0.99**	**0.99**	0.98	0.80	0.88	0.94	0.94	0.94	0.97	0.97	0.97
D4	**0.95**	**0.95**	**0.95**	0.88	0.81	0.85	0.92	0.90	0.91	0.90	0.86	0.88
D5	**0.97**	**0.96**	**0.96**	0.87	0.74	0.80	0.92	0.89	0.91	0.89	0.85	0.87
D6	**0.99**	**0.99**	**0.99**	0.97	0.95	0.96	0.93	0.91	0.92	0.93	0.92	0.93
D7	**0.99**	**0.99**	**0.99**	0.90	0.80	0.85	0.79	0.81	0.80	0.94	0.88	0.91
H.Mean	**0.97**	**0.97**	**0.97**	0.92	0.80	0.85	0.89	0.88	0.89	0.93	0.92	0.92

total while the first two steps and the instance matching cover only 21.32% and 24.23%, respectively. In general the speed of SLINT+ is very high when it takes below 10 minutes to interlink the largest subset with 37,120 links and nearly 1 second for the smallest one with 1,008 links.

4.4 Comparison with previous systems

We compare SLINT+ with AgreementMaker [3], SERIMI [1], and Zhishi.Links [8]. These systems recently participated the OAEI 2011 Instance Matching Campaign [4]. Among these systems, SERIMI is the only one that is domain-independent while AgreementMaker and Zhishi.Links are not. Table 6 shows the results of SLINT+ and the others. According to this table, SLINT+ is very much higher if compared with other systems on both precision and recall. Zhishi.Links is the second best in this comparison but is still 0.05 lower than SLINT+ in overall. Particularly, SLINT+ performs very well on D4 and D5 datasets, which are seem to be difficult for Zhishi.Links to interlink.

The improvement of SLINT+ against compared systems is the confirmation of the robustness of domain-independent approach over domain-dependent ones, such as AgreementMaker and Zhishi.Links. The limitation of SERIMI is that this system solely uses the string measurement and only focus on short strings.

Therefore, this system is lower in precision and recall than those of SLINT+ on every subset.

5 Conclusion

In this paper, we present SLINT+, a domain-independent linked data interlinking system. Our system interlinks different data sources by comparing the instances using collective information and weighted similarity measure. The information to be compared are extracted by RDF predicates which are automatically selected using their covering and discriminative abilities. The key idea to remove the requirement of labeled matched instances is the concept of the confidence of a predicate alignment. The candidate generation step is also investigated and evaluated. Experimental results show that SLINT+ is outstanding comparing to previous famous systems on OAEI2011 dataset. The domain-independent capability can be said to be achieved when SLINT+ perform very well on 119 different domains of data.

In the future, we will study on improving the scalability of SLINT+ in order to match very large data sources. Besides, a cross-domain interlinking system, which aims at matching the sources containing multiple domains, is a very interesting objective to be explored.

References

1. S. Araujo, D. Tran, A. de Vries, J. Hidders, and D. Schwabe. SERIMI: Class-based disambiguation for effective instance matching over heterogeneous web data. In *SIGMOD'12 15th Workshop on Web and Database*, pages 19–25, 2012.
2. C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Semantic Web and Information Systems*, 4(2):1–22, 2009.
3. I. F. Cruz, F. P. Antonelli, and C. Stroe. AgreementMaker: efficient matching for large real-world schemas and ontologies. *VLDB Endow.*, 2:1586–1589, 2009.
4. J. Euzenat, W. Ferrara, A. Hage, L. Hollink, C. Meilicke, A. Nikolov, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Zamazal, and C. Trojahn. Final results of the ontology alignment evaluation initiative 2011. In *ISWC' 11 6th Workshop on Ontology Matching*, pages 85–113, 2011.
5. R. Isele and C. Bizer. Learning linkage rules using genetic programming. In *ISWC' 11 6th Workshop on Ontology Matching*, pages 13–24, 2011.
6. K. Nguyen, R. Ichise, and B. Le. Learning approach for domain-independent linked data instance matching. In *KDD'12 2nd Workshop on Minning Data Semantics*, pages 7:1–7:8, 2012.
7. K. Nguyen, R. Ichise, and B. Le. SLINT: A schema-independent linked data interlinking system. In *ISWC'12 7th Workshop on Ontology Matching*, 2012.
8. X. Niu, S. Rong, Y. Zhang, and H. Wang. Zhishi.links results for OAEI 2011. In *ISWC' 11 6th Workshop on Ontology Matching*, pages 220–227, 2011.
9. D. Song and J. Heflin. Automatically generating data linkages using a domain-independent candidate selection approach. In *ISWC' 11*, pages 649–664, 2011.
10. J. Volz, C. Bizez, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *ISWC' 09*, pages 650–665, 2009.