SLINT: A Schema-Independent Linked Data Interlinking 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 all instances that represent the same real-world object and locate in different data sources. Since different data publishers frequently use different schemas for storing resources, we aim at developing a schema-independent interlinking system. Our system automatically selects important predicates and useful predicate alignments, which are used as the key for blocking and instance matching. The key distinction of our system is the use of weighted co-occurrence and adaptive filtering in blocking and instance matching. Experimental results show that the system highly improves the precision and recall over some recent ones. The performance of the system and the efficiency of main steps are also discussed.

Keywords: linked data, schema-independent, blocking, interlinking.

1 Introduction

Years of effort in building linked data has brought a huge amount of data in the LOD. However, maximizing the efficiency of linked data in the development of semantic web is still facing many difficulties. One of the current challenges is to integrate the individual data sources for building a common knowledge system. When different data source may contain heterogeneous instances, which co-refer to the same real-world objects, data integration process requires the detection of such objects to ensure the integrity and consistency of data. Detecting all identities between data sources is the mission of data interlinking. Data interlinking consist of two main steps, blocking and instance matching. While blocking aims at pruning the number of comparison, instance matching is to determine the matching status of two interested instances.

Current interlinking methods can be categorized into two main groups: schema-dependent [2,7,10] and schema-independent [1,3,4,9]. The former requires the knowledge about meaning of RDF predicates (e.g. predicate #preLabel declares the label of object) and the predicate alignments (e.g. predicate #preLabel matched with predicate #name). In contrast, the latter does not need these information, therefore it does not rely on human knowledge about the schema. Because a linked data instance is a set of many RDF triples (subject, predicate, object), the schema of a data source refers to the list of all used predicates, which are closely related to vocabulary and ontology. The schemas are usually different for each data sources, even in the same data source but different domains. Clearly, schema-independent methods are more applicable when it can

work on every kind of source or domain without any human's instruction. Besides, manual specifications of interlinking rules frequently ignore the hidden useful predicate alignments.

We present SLINT system, which use a new approach for schema-independent linked data interlinking. SLINT automatically selects important RDF predicates using the coverage and discriminability. The selected predicates are combined to construct the predicate alignments in conciliation of data type. We estimate the confidence of predicate alignments to collect the most appropriate alignments for blocking and interlinking. By this way, the collective information of instance is frequently leveraged. Blocking is therefore more complete, compact, and supportive for interlinking. Also, we apply adaptive filtering techniques for blocking and instance matching. In experiment, we compare SLINT with three systems, which participated OAEI 2011 instance matching campaign, and report the high improvement on both precision and recall. Experiments on the performance of SLINT and the efficiency of blocking step are also reported.

The paper is organized as follow: the next section is the overview of previous work. Section 3 describes the detail of SLINT system. Section 4 reports our experimental evaluation. Section 5 closes the paper with conclusion and outlook.

2 Related work

Data interlinking is an early studied area, however, this problem in linked data has just been recently attended. Silk [10], a well-known framework, provides a declarative interface for user to define the predicate alignments as well as the similarity metrics for matching. Silk was used as a main component of the LDIF [8], a multiple linked data sources integration framework. Recently, Isele and Bizer have improved their Silk by applying an automatic linkage rules generation using genetic algorithm [3]. The work is very interesting at the modeling of appropriate fitness function and specific transformations for genetic programming in the context of interlinking. This work makes Silk to be schema-independent. With the similar objective, RAVEN [4] minimize human curation effort using active learning, while Nikolov et al. also use genetic algorithm with the research target is an unsupervised learning process [6]. Also with schema-independent goal, Nguyen et al. suggest using decision tree classifier for determining the matching status of two instances [5].

Zhishi.Links [7] is one of the current state-of-the-art matchers. This system adopt the idea of Silk's pre-matching step, by using label of objects such as *skos:prefLabel* or *scheme:label*, to group similar instances. Afterward, a more complex semantic similarity is utilized for matching. This system ranks first at the OAEI 2011 instance matching, while the second best is SERIMI [1], a schema-independent system. SERIMI selects RDF predicates and predicate alignments using entropy and RDF object similarity, respectively. AgreementMaker [2] is an ontology matching and instance matching system. It firstly generates candidates set by comparing the labels of instances. These candidates are then divided into smaller subsets, in which every pair is matched to produce the final alignments.

Most of previous interlinking systems do not deeply investigate on blocking step, which generates potential identity pairs of instances. Song and Heffin focus

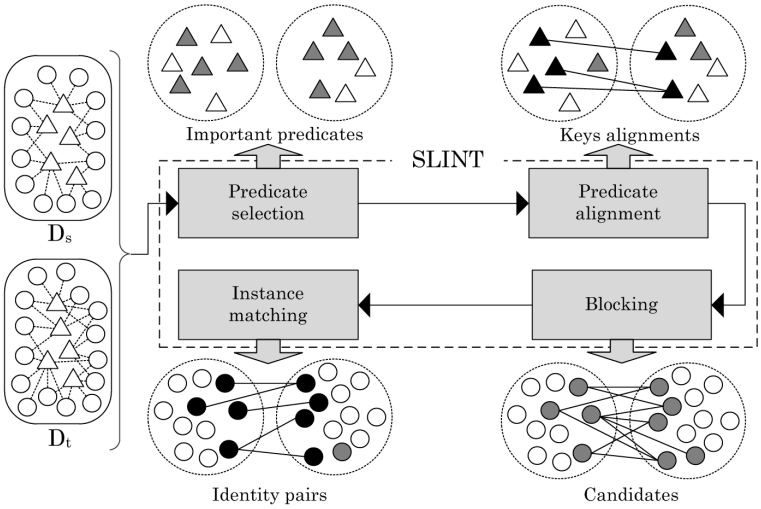

Fig. 1. Summary of data interlinking process

on blocking scheme for linked data interlinking for parallel independent work [9]. It is a very interesting idea when the authors propose an unsupervised learning for maximizing the usefulness of blocking keys, which are the combinations of RDF predicates. The authors conduct experiments on some large datasets, which also proof for the scalability.

In general, the schema-dependent approaches compare two instances by specified properties. That is, they can detect almost right identity pairs but the precision may be low on highly ambiguous data sources. The reason is that some useful information can be ignored since the manual predicate alignment frequently is not an optimal solution. In contrast, schema-independent approaches reconcile precision and recall because of elaborate analysis on the data. Although these approaches need to collect predicate alignments, the matching is more effective when collective information is frequently used. Comparing SLINT with previous interlinking systems, the prominent differences are the predicate selection, predicate alignment, and adaptive filtering for blocking and interlinking. In the next section, we describe these elements as the details of SLINT.

3 Schema-independent linked data interlinking system

This section describes the SLINT system. The overview of the interlinking process for source data D_s and target data D_t is shown in Figure 1. In this figure, the small circles and triangles respectively stand for instances and theirs RDF predicates. The referred circles of each step are the output of that step. The SLINT system consists of four steps. The interlinking process begins with *predicate selection*, which collects the *important predicates* from all predicates of each data sources. In the second step, *predicate alignment*, selected predicates are combined in accordance with their data type to construct the raw predicate alignments. We estimate the confidence of every raw alignment to measure its appropriateness. A raw alignment will be called a key alignment if its confidence

satisfies a filtering condition. These *key alignments* provide much useful information in blocking and instance matching steps. Next, *blocking* step is designed to reduce the number of comparison by producing identity *candidates* of instances. The *instance matching* afterward only need to verifies the retrieved candidates for discovering the *identity pairs*. The followings are the detail of each step.

3.1 Predicate selection

The mission of this step is to find the important predicates from the schema, which consists of all predicates appearing in interested data source. We use two criteria for determining the importance level of predicate p: coverage $cov(p, D)$ and discriminability $dis(p, D)$. Eq.1 and Eq. 2 are the explanations of these criteria when considering predicate p of data source D.

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

$$dis(p, D) = \frac{|\{o|\exists x \in D, < s, p, o >\in x\}|}{|\{< s, p, o > |\exists x \in D, < s, p, o >\in x\}|}. \quad (2)$$

In these equation, x represents an instance and is a set of RDF triple $< s, p, o >$ (subject, predicate, object). D is the interested data source and is a set of instances. From each input source, we collect the predicates having high score of coverage and discriminability. A predicate p is selected if it satisfies the condition in Eq.3, which inherits from the idea of [9].

$$(cov(p, D) \geq \alpha) \wedge (dis(p, D) \geq \beta) \wedge (HMean(cov(p, D), dis(p, D)) \geq \gamma). \quad (3)$$

The α and β imply the minimum standard of an important predicate, whereas γ, the condition for harmonic mean of $dis(p, D)$ and $cov(p, D)$, is the main requirement. Therefore, we set small values for α and β and larger value for γ.

Song and Heffin focus on learning blocking key by iteratively maximize the coverage and discriminability of the set of predicates [9]. In our system, we use the same discriminability function with theirs and slightly different function for coverage. For the numerator of Eq. 1, while they use the number of RDF subjects, we use the number of instances, because we aim at finding the frequency of predicate over instances, not over RDF subjects.

Important predicates are expected to be used for declaring the common properties and distinct information of objects. Since coverage and discriminability respectively express the former and latter, the combination of them is therefore appropriate for the objective of predicate selection. If a predicate has a high coverage but a low discriminability or otherwise, it will not be important. An example for this kind of predicate is *rdf:type*. This predicate is frequently used but it usually describes a limit range of various RDF objects when observing the instances in the same domain.

3.2 Predicate alignment

In this step, we find the appropriate alignments of predicates between the source data and target data. An alignment of two predicates is considered to be appropriate if the interested predicates describe the similar properties of instances.

From selected predicates of source data and target data, we connect every type-matched pair and select the alignments whose confidence is higher than threshold δ. Selected predicate alignments are called key alignments. The confidence of an alignment is the Dice coefficient between the representatives of RDF objects described by its formed predicates. Eq. 4 is the equation of confidence $conf(p_s, p_t)$ for the alignment between predicate p_s in source data D_s and predicate p_t in target data D_t.

$$conf(p_s, p_t) = \frac{2 \times |R(O_s) \cap R(O_t)|}{|R(O_s)| + |R(O_t)|}, O_k = \{o | \exists x \in D_k, < s, p_k, o > \in x\}. \quad (4)$$

In above equation, R is the function that returns the representative elements of RDF objects. The return values of R depend on the type of predicates. We divide the predicates into five different types: *string*, *URI*, *decimal*, *integer*, and *date*. This separation is based on the variety of data types in the real world and covers most of current types of linked data. For *string*, we extract the word token of RDF objects. For *URI*, we omit the domain part and use the same manner as for *string*, with the assumption that slash '/' is the token separator. For *decimal*, we take the 2-decimal digits rounded values. For *integer* and *date*, we do not transform the RDF objects and use the original values. For determining type of a predicate, we use the major type of RDF objects declared by this predicate. For example, if 51% appearance times of p is to describe *decimal* values, the data type of p will be *decimal*. Currently, we detect the type of RDF objects without the consideration about the difference in their metric (e.g. the units of time, distance, area).

The confidence of an alignment represents the similarity of RDF objects between two data sources. The predicates having the same meaning frequently describe the similar information. Therefore, alignments of matched predicates usually have higher confidence than the others. It means that a predicate satisfying the requirements of an important predicate is verified again, by considering the confidence of all alignments in which it appears. For example, the predicate *rdfs:comment* has possibility to be important but the confidences of its alignments are usually low because the denominator of Eq.3 is very high in this case.

A common limiting point of almost previous systems is the use of string measurement for every type of RDF objects. Clearly, this approach is not sufficient to cover the meaning of RDF objects, thus, does not well estimate the similarity of non-string values. We discriminate data types not only in combining predicates, but also in blocking and instance matching.

It is not easy for a person to detect all useful predicate alignments, this step is therefore very meaningful, and in accompaniment with predicate selection, it tackles the schema-independent goals. The next steps are the use of selected key alignments and their formed predicates.

3.3 Blocking

As we introduced, the blocking aims at retrieving the candidates for instance matching step by grouping similar instances into the same block. A candidate is a pair of two instances, one belongs to source data and one belongs to target data. The blocking can be divided into three sub phases. The first phase indexes

Algorithm 1: Generating candidates set

Input: D_s, D_t, Pr_s, Pr_t, ζ, ϵ
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** $< x_s, v_s > \in H.GetValues(key, D_s)$ **do**
14 **foreach** $< x_t, v_t > \in H.GetValues(key, D_t)$ **do**
15 $M[x_s, x_t] \leftarrow M[x_s, x_t] + v_s \times v_t$
16 **foreach** $x_s \in D_s$ **do**
17 **foreach** $x_t \in D_t$ **do**
18 $max_s \leftarrow \text{Max}(M[x_s, x_j]), \forall x_j \in D_t$
19 $max_t \leftarrow \text{Max}(M[x_i, x_t]), \forall x_i \in D_s$
20 $max \leftarrow \text{HMean}(max_s, max_t)$
21 **if** $M[x_s, x_t] \geq \zeta$ *and* $\frac{M[x_s, x_t]}{max} \geq \epsilon$ **then**
22 $C \leftarrow C \cup < x_s, x_t >$
23 **return** C

every instance in each data source by the value extracted from its RDF objects. The second phase traverses the index table and builds a weighted co-occurrence matrix. The final phase uses this matrix as the input information when it applies a filtering technique to select candidates. Algorithm 1 is the pseudo-code of the whole blocking process. In this algorithm, Pr_s and Pr_t represent the list of predicates that form the key alignments, where Pr_k belongs to D_k. H, M, C, Rp represent the inverted-index table, weighted co-occurrence matrix, candidates set, and representative extraction method, respectively.

The lines 4-11 perform the invert-indexing, a well-known indexing technique. By once traversing each data source, we extract the representatives of RDF objects and use them as the keys of invert-index table. An element r in the representatives set of RDF objects consists of two fields: the label $r.Label$ and value $r.Value$. While $r.Label$ is the return value of representative extraction

method R as in predicate alignment step, $r.Value$ is computed in accordance with the data type of predicate p_i. If p_i is *string* or *URI*, we set the value to TF-IDF score of the token. If p_i is either *decimal*, *integer*, or *date*, we assign the value to a fixed number, which is 1.

After constructing the invert-index table, we compute weighted co-occurrence matrix M as the lines 12-15, by accumulating the value for each matrix element.

The lines 16-22 are the process of adaptive filtering. An instance pair $< x_s, x_t >$ will be considered as a candidate if its weighted co-occurrence value $M[x_s, x_t]$ satisfies the threshold ϵ, after divided for the harmonic mean of maximum weighted co-occurrences of x_s and x_t. In addition, we use ζ, a small threshold, to avoid the surjection assumption. The identities frequently have the high weighted co-occurrences; however, these values are variable for different pairs. Choosing a fixed threshold for selecting candidates is not good in this situation and is a tedious task. Therefore, we use the coefficient of $M[x_s, x_t]$ and max, which is a data driven element and expresses the adaptive filtering idea.

Blocking is very important because it reduces the number of comparison in instance matching. However, it seems not to have been sufficiently attended when most of previous systems use quite simple method for blocking. In comparison with blocking step in previous interlinking systems, the key difference of our method is the weighted co-occurrence matrix and the adaptive filtering. While previous systems compare the pairs of RDF objects, we aggregate the product of the weight of their matched representatives. For candidate selection, Silk [10] and Zhishi.Links [7] use *top-k* strategy, which selects k candidates for each instance. The approach is very good for controlling the number of candidates, but determining the value of k is not easy. Song and Heffin use *thresholding* selection [9], which is also similar with SERIMI [1]. Our method also use thresholding approach as the availability of ζ. However, the key idea of our selection method is the adaptive filtering because the impact of ζ is not high. Frequently, there are many of non-identity pairs between two data sources, ζ is therefore usually configured with a low value.

Next, we input the set of candidates C and the key alignments A into the final step, instance matching.

3.4 Instance matching

The instance matching verifies the selected candidates to determine their identity state. We compute the matching *score* for every candidate and choose the ones that have high score as the identity pairs. For each element in A, we compute the similarity of RDF objects, which declared by the involved predicates of interested key alignment. The final score of two instances is the weighted average value of all these similarities, and the weights are the confidences of key alignments. Eq.5 is the computation of matching score between instance $x_s \in D_s$ and $x_t \in D_t$.

$$\begin{aligned} score(x_s, x_t) =& \frac{1}{W} \sum_{<p_s, p_t> \in A} conf(p_s, p_t) \times sim(R(O_s), R(O_t)), \\ Where \quad & O_k = \{o | \exists x \in D_k, < s, p_k, o > \in x\} \\ & W = \sum_{<p_s, p_t> \in A} conf(p_s, p_t). \end{aligned} \tag{5}$$

In this equation, R stands for the representative extraction methods, which are similar to those in predicate alignment step.

Categorizing five data types, we implement three different versions of *sim* function in accordance with the type of predicates. For *decimal* and *integer*, we take the variance of the values to remedy the slight difference of data representations. For *date*, the *sim* function yields 1 or 0 when the values are equal or not, respectively. A date is usually important if it is a property of the object (e.g. birthday, decease date, release date of a movie). Therefore, the exact matching is an appropriate selection for dates comparison. For *string* and *URI*, we compute the TF-IDF modified cosine similarity, as given in Eq.6. TF-IDF is used because its advantage in disambiguating the instances sharing common tokens. TF-IDF also minimizes the weight for the stop-words, which are usually useless.

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

Similar with blocking step, we do not use fixed single threshold for filtering the candidates. Two instances will be considered as an identity pair if their score is higher than the maximum score of the candidates in which either of instances appears. The final identities set I is formalized in Eq.7.

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

An identity pair is expected to be the highest correlated candidate of each instance. However, it usually is not true because the ambiguity of instances. A thresholding method that relies on the highest score would be better in this situation. While true identity pair and its impostors have the similar score, θ is assigned with a quite large value. On the other hand, η is additionally configured as the minimum requirement of an identity. Like ϵ in Algorithm 1, η ensures there is no assumption about the surjection of given data sources.

The key distinctions of our approach in comparison with the previous are the use of weighted average and adaptive filtering. Previous systems do not have the data driven information like confidence of key alignments. Silk [10] provides a manual weighting method; however, a good weight usually depends much on the human knowledge about the data. For identity selection, Zhishi.Links [7] and AgreementMaker[2] eventually select the best correlated candidates, while Silk [10] and SERIMI [1] use threshold-based selection. We compare the interlinking result of our system with those of Zhishi.Links, SERIMI, and AgreementMaker in our experiments, which are reported in the next section.

4 Experiment

4.1 Experiment setup

We evaluate the efficiency of blocking step and the whole interlinking process of SLINT. We also compare SLINT with Zhishi.Links [7], SERIMI [1], and

AgreementMaker [2], which recently participated OAEI 2011 instance matching campaign. Discussion on predicate selection and predicate alignment are also included. For every test in our experiment, we use the same value for each threshold. We set α, β ,γ (Eq. 3), δ (Eq. 4), ζ, ϵ (Algorithm 1), η, and θ (Eq. 7) to 0.25, 0.25, 0.5, 0.25, 0.1, 0.5, 0.25, and 0.95, respectively. The fixed values of α, β, γ and δ express the schema-independent capability of SLINT.

Like previous studies, for blocking, we use two evaluation metrics: pair completeness (PC) and reduction ratio (RR); For interlinking, we use recall (Rec), precision (Prec), and F1 score, the harmonic mean of recall and precision. Eq.8 and Eq.9, Eq.10, and Eq.11 are the computations of used metrics.

$$PC = \frac{Number\ of\ correct\ candidates}{Number\ of\ actual\ identity\ pairs}. \tag{8}$$

$$RR\ =\ 1\ -\ \frac{Number\ of\ candidates}{Number\ of\ all\ pairs}. \tag{9}$$

$$Rec\ =\ \frac{Number\ of\ correct\ identity\ pairs}{Number\ of\ actual\ identity\ pairs}. \tag{10}$$

$$Prec\ =\ \frac{Number\ of\ correct\ identity\ pairs}{Number\ of\ discovered\ pairs}. \tag{11}$$

The performance of an interlinking system is also very important. We report the execution times of SLINT when running on a desktop machine equipped with 2.66Ghz quad-core CPU and 4GB of memory.

4.2 Datasets and discussion on predicate selection & predicate alignment

We use 9 datasets in experiment. The first 7 datasets are IM@OAEI2011 datasets, the ones used in instance matching campaign at the OAEI 2011[3]. Concretely, we use the datasets of interlinking New York Times track, which asks participants to detect identity pairs from NYTimes to DBpedia, Freebase, and Geonames. These datasets belong to three domains: *locations*, *organizations*, and *people*. The IM@OAEI2011 datasets are quite small. Therefore, for evaluating the computational performance of the system, we select two larger datasets. The first one, a medium dataset, contains 13758 pairs in *film* domain between DBpedia and LinkedMDB[4]. The second one is a quite large dataset, which contains 86456 pairs in *locations* domain between DBpedia and Geonames[5]. All datasets are downloaded by dereferencing URI and stored in external memory in advance. We remove triples having *owl:sameAs* predicates and *rdf:seeAlso* predicates of course. Table 1 gives the overview of these datasets. In this table, IM@OAEI2011 datasets are from D1 to D7, and the last two datasets are D8 and D9. We also include the number of predicates and predicate alignments in this table. Denotes that s and t are the source and target data, respectively; Pr_d and Pf_d are the

[3] http://oaei.ontologymatching.org/2011/instance/
[4] http://downloads.dbpedia.org/3.7/links/linkedmdb_links.nt.bz2
[5] http://downloads.dbpedia.org/3.7/links/geonames_links.nt.bz2

Table 1. Number of predicates and predicate alignments

ID	Source	Target	Domain	Pairs	Pr_s	Pr_t	Pf_s	Pf_t	A	K
D1	NYTimes	DBpedia	Locations	1920	12	2859	6	24	26	7
D2	NYTimes	DBpedia	Organizations	1949	10	1735	6	14	21	5
D3	NYTimes	DBpedia	People	4977	10	1941	5	20	33	8
D4	NYTimes	Freebase	Locations	1920	12	775	6	18	20	4
D5	NYTimes	Freebase	Organizations	3044	10	1492	5	18	33	3
D6	NYTimes	Freebase	People	4979	10	1844	5	18	32	5
D7	NYTimes	Geonames	Locations	1789	12	32	6	13	14	4
D8	DBpedia	LinkdMDB	Movie	13758	1343	54	16	7	31	14
D9	DBpedia	Geonames	Locations	86456	8194	17	11	7	27	8

number of predicates in data source d before and after selected by predicate selection step, respectively; A and K are the number of all predicate alignments and only key alignments, respectively.

In general, excepts in NYTimes, the number of available predicates in the schema of each data source is very large, but the important predicates occupy a very small percent. As our observation, the predicates declaring the label or the name of objects are always aligned with a very high confidence. The non-string type predicates also frequently construct the key alignments. For example, in *locations* domain, the key alignments always contain the right combination of predicates declaring latitudes and longitudes. The predicate *releaseDate* of DBpedia is successfully combined with predicate *date* and predicate *initial_relase_date* of LinkedMDB. An interesting key alignment in dataset D6 is the high confidence combination of *core#preLabel* of NYTimes and *user.mikeshwe.default_domain.videosurf_card.videosurf_link_text* of Freebase. The latter predicate may be difficult for manual selection since the meaning of the predicate name does not imply the label. Clearly, it is not easy for human to detect every compatible predicates. When manually doing this task, we may lose to leverage all useful information.

4.3 Blocking and interlinking result

This section reports the result of blocking and the whole interlinking process. Concretely, we report the pair completeness and reduction ratio of blocking, and precision, recall, F1 score, and the runtime of the system. Table 2 shows these metrics on each dataset. According to this table, although we cannot retain all identity pairs, the lowest PC is still very high at 0.94. Besides, the high RRs reveal that the numbers of retrieved candidates are very small if compared with the numbers of total pairs. For all the evidences of PC and RR, the aim of blocking is successfully achieved.

For interlinking, the precison and recall are very competitive. The recall, which is not much lower than pair completeness, implies that the instance matching performs a good work. The high precision implies that our system has a efficient disambiguation capability on tested datasets. It seems easy for SLINT to interlink *people* domain, whereas in *locations* domain, SLINT achieves the best result on IM@OAEI2011 datasets involving with Geonames.

The execution time of SLINT is very good in overview. Because we use co-occurrence structure in blocking, the memory on tested machine cannot satisfy

Table 2. Number of candidates, PC, RR, Rec, Prec, F1, and execution time

Dataset	Blocking			Interlinking			
	Candidates	PC	RR	Prec	Rec	F1	Runtime
D1	4102	0.9901	0.9989	0.9636	0.9651	0.9644	3.55
D2	3457	0.9831	0.9970	0.9768	0.9487	0.9625	4.29
D3	9628	0.9950	0.9972	0.9883	0.9841	0.9862	12.74
D4	3580	0.9849	0.9990	0.9486	0.9521	0.9504	3.78
D5	7744	0.9823	0.9992	0.9610	0.9560	0.9585	6.71
D6	10333	0.9938	0.9996	0.9944	0.9904	0.9924	18.25
D7	2473	0.9961	0.9959	0.9883	0.9888	0.9885	1.63
D8	33926	0.9948	0.9998	0.9317	0.9868	0.9584	67.76
D9	418592	0.9468	0.9999	0.9782	0.9418	0.9596	2465.38

a very large dataset. In our context, interlinking dataset D9 has such issue. We temporarily implement a parallel program for re-computing every element of the co-occurrence matrix. The interlinking on this dataset is therefore takes much time because the repeat of data traversing and high computational cost. However, the high speeds on other datasets are really promising for scaling-up our system.

The advantage of blocking is very high if we compare the time of interlinking with and without this step. For example, the time for instance matching step to match 33926 candidates of dataset D8 is 12.2 seconds. It means that the time for matching all available pairs will be nearly 17 hours, whereas this number is only 67.76 seconds in total if we implement the blocking step. Blocking averagely occupies 58% total runtime of interlinking process on the nine tested datasets. Although this number is over a half, the advantage of blocking is still very considerable.

4.4 Comparison with previous interlinking systems

As mentioned, we compare our system with AgreementMaker [2], SERIMI [1], and Zhishi.Links [7]. Because these systems recently participated instance matching campaign of the OAEI 2011, we use the results on IM@OAEI2011 datasets for comparison. Table 3 shows the interlinking result of SLINT and others. As showed in this table, it is clear that our system totally outperforms the others on both precision and recall. AgreementMaker has a competitive precision with SLINT on dataset D3 but this system is much lower in recall. Zhishi.Links results on dataset D3 are very high, but the F1 score of SLINT is still 0.05 higher in overall.

The prominent differences of SLINT and these systems are that we use the confidence of alignment as the weight for blocking and instance matching, and discriminate data types with the use of TF-IDF for the token of string and URI. Generally, SLINT is verified as the best accurate one among compared systems.

5 Conclusion

In this paper, we present SLINT, an efficient schema-independent linked data interlinking system. We select important predicates by predicate's coverage and discriminability. The predicate alignments are constructed and filtered for obtaining key alignments. We implement an adaptive filtering technique to produce

Table 3. Comparison with previous interlinking systems.

Dataset	SLINT			Agree.Maker			SERIMI			Zhishi.Links		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
D1	**0.96**	**0.97**	**0.96**	0.79	0.61	0.69	0.69	0.67	0.68	0.92	0.91	0.92
D2	**0.98**	**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**	**0.98**	**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	**0.96**	**0.96**	**0.96**	0.87	0.74	0.80	5.92	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

candidates and identities. Compare with the most recent systems, SLINT highly outperforms the precision and recall in interlinking. The performance of SLINT is also very high when it takes around 1 minute to detect more than 13,000 identity pairs.

Although SLINT has good result on tested datasets, it is not sufficient to evaluate the scalability of our system, which we consider as the current limiting point because of the used of weighted co-occurrence matrix. We will investigate about a solution for this issue in our next work. Besides, we also interested in automatic configuration for every threshold used in SLINT and improving SLINT into a novel cross-domain interlinking system.

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. 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.
3. R. Isele and C. Bizer. Learning linkage rules using genetic programming. In *ISWC' 11 6th Workshop on Ontology Matching*, pages 13–24, 2011.
4. A. Ngomo, J. Lehmann, S. Auer, and K. Höffner. RAVEN - Active learning of link specifications. In *ISWC' 11 6th Workshop on Ontology Matching*, pages 25–36, 2011.
5. K. Nguyen, R. Ichise, and B. Le. Learning approach for domain-independent linked data instance matching. In *KDD'12 2nd Workshop on Minning Data Semantic*, pages 7:1–7:8, 2012.
6. A. Nikolov, M. d'Aquin, and E. Motta. Unsupervised learning of link discovery configuration. In *ESCW'12*, pages 119–133, 2012.
7. 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.
8. A. Schultz, A. Matteini, R. Isele, C. Bizer, and C. Becker. LDIF - Linked data integration framework. In *ISWC' 11 2nd Workshop on Consuming Linked Data*, 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.