Shao C, Hu LM, Li JZ *et al.* RiMOM-IM: A novel iterative framework for instance matching. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 31(1): 185–197 Jan. 2016. DOI 10.1007/s11390-016-1620-z

RiMOM-IM: A Novel Iterative Framework for Instance Matching

Chao Shao [1], Lin-Mei Hu [1,*], Juan-Zi Li [1], *Member, CCF*, Zhi-Chun Wang [2], *Member, CCF*
Tonglee Chung [1], and Jun-Bo Xia [1]

[1] *Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*
[2] *College of Information Science and Technology, Beijing Normal University, Beijing 100875, China*

E-mail: {birdlinux, hulinmei1991, lijuanzi2008}@gmail.com; zcwang@bnu.edu.cn; thdonglai@163.com
shiningbeer@gmail.com

Received October 30, 2014; revised July 17, 2015.

Abstract Instance matching, which aims at discovering the correspondences of instances between knowledge bases, is a fundamental issue for the ontological data sharing and integration in Semantic Web. Although considerable instance matching approaches have already been proposed, how to ensure both high accuracy and efficiency is still a big challenge when dealing with large-scale knowledge bases. This paper proposes an iterative framework, RiMOM-IM (RiMOM-Instance Matching). The key idea behind this framework is to fully utilize the distinctive and available matching information to improve the efficiency and control the error propagation. We participated in the 2013 and 2014 competition of Ontology Alignment Evaluation Initiative (OAEI), and our system was ranked the first. Furthermore, the experiments on previous OAEI datasets also show that our system performs the best.

Keywords instance matching, large-scale knowledge base, blocking, similarity aggregation

1 Introduction

Recently, a number of ontological knowledge bases have been built and published, such as DBpedia①, YAGO② and Xlore③. Some published knowledge bases are domain-specific ones which contain facts within one domain, such as movie, music and geography; some others are cross-domain knowledge bases that consist of various kinds of information in different domains. Usually, the knowledge about one object can be contained in different knowledge bases. For example, both YAGO and elvisPedia contain the information about a person named "Elvis Presley". YAGO records the birthdate of this person while elvisPedia has the information about his wife. If we want to know more about "Elvis Presley", we have to search his information in different knowledge bases. Therefore, there is a growing need to align a knowledge base so that we can easily get more complete understanding about things that we are interested in.

A lot of work has already been done for aligning ontological knowledge bases. Early research focuses on aligning schema elements (i.e., concepts and properties) in knowledge bases, which is called ontology matching. Recently, the problem of matching instances in knowledge bases has attracted increasing attention. Many instance matching approaches have been proposed. However, most of them cannot deal with large-scale knowledge bases nicely because they require traversing all instance pairs between two knowledge bases[1-3]. Some other approaches, such as CODI[4], Silk[5], PARIS[6],

Regular Paper

The work is supported by the National Basic Research 973 Program of China under Grant No. 2014CB340504, the National Natural Science Foundation of China and the French National Research Agency under Grant No. 61261130588, the Tsinghua University Initiative Scientific Research Program under Grant No. 20131089256, the Science and Technology Support Program of China under Grant No. 2014BAK04B00 and the Tsinghua University and National University of Singapore Extreme Search Joint Centre.

*Corresponding Author

①http://dbpedia.org/About, Nov. 2015.

②http://www.mpi-inf.mpg.de/yago-naga/yago/, Nov. 2015.

③http://xlore.org/index, Dec. 2015.

and SIGMa[7], are proposed for large-scale instance matching. There are two major techniques in existing approaches to speed up the instance matching process: blocking and iterative matching. Blocking indexes the instances in two knowledge bases separately and then selects the instances in multiple knowledge bases but with the same key as candidate instance pairs. Iterative matching finds the instance correspondences in multiple loops; only a fraction of instances are matched in each iteration, which are then used as seeds for matching the remaining instances in the next iterations. Although the two techniques above are very helpful to large-scale instance matching, there are still several challenging problems which are not well addressed. First, since only literal values in RDF triples are used as the indexing keys for blocking usually, the set of candidate instance pairs is still very large. Second, iterative instance matching is likely to propagate minor errors of mismatched instances in each iteration. Traditional decision-making methods can hardly get rid of mismatched instances since instances in two different knowledge bases are usually described by different numbers of RDF triples.

In order to solve the above challenges in large-scale instance matching, we propose an iterative instance matching framework RiMOM-IM (RiMOM-Instance Matching), which is developed based on our ontology matching system RiMOM[8]. The main idea behind the framework is to maximize the utilization of distinctive and available matching information. RiMOM-IM presents a novel blocking method to improve the efficiency and employs a weighted exponential function based similarity aggregation method to guarantee the high accuracy of instance matching. Specifically, the main contributions of our work are summarized as follows.

1) We propose and develop a customizable iterative instance matching framework RiMOM-IM, which can efficiently handle large-scale instance matching tasks. RiMOM-IM matches instances in an iterative way, which utilizes the aligned instances for matching the remaining instances in each iteration.

2) In the proposed framework, a new blocking method is proposed to select candidate instance pairs. Our method uses predicates and their distinctive object features as keys to index the instances, which is then used to select candidate instance pairs. Our blocking method can effectively reduce the running time without decreasing the precision and recall.

3) A weighted exponential function based aggregation method *ExpAgg* for similarity aggregation is proposed to draw the matching results from the similarities between instances in terms of different predicates. *ExpAgg* can produce accurate results even when the aligned predicate numbers among different pairs vary a lot.

4) Extensive experiments are conducted to evaluate our proposed approach under the principled means of evaluation. Results on benchmark datasets of OAEI@IM④ demonstrate that our framework significantly outperforms state-of-the-art approaches in both accuracy and efficiency. Our system can match two knowledge bases with over 6 million facts within 30 minutes on a desktop machine equipped with 3.3 Ghz quad-core CPU and 4 GB memory.

The rest of the paper is organized as follows. In Section 2, we formalize the problem definition with some notation declarations. We describe the framework of RiMOM-IM in detail in Section 3. Experiments and analysis are given in Section 4. In Section 5, we review related work. We conclude the paper in Section 6.

2 Preliminaries

In this section, we present some concepts that will be used through the paper as follows.

Knowledge Base. A knowledge base KB is a tuple (I, P, L, F) where I, P, L are sets of instances, predicates and literals. $F \subseteq I \times P \times (I \cup L)$, denotes a set of facts, $\{(s, p, o) \mid s \in I,\ p \in P,\ o \in I \cup L\}$, stored in RDF triple set.

Instance Matching. Given two knowledge bases KB_{S} and KB_{T}, the instance matching from KB_{S} to KB_{T} is to find *owl: sameAs* relation between instances in the source knowledge base instance set I_{S} and the target knowledge base instance set I_{T}, where KB_{S} and KB_{T} are the source and the target knowledge bases respectively. In this work, we suppose each instance in KB_{T} can match at most one instance of KB_{S}, and every instance in KB_{S} may be aligned with many instances in KB_{T}.

In fact, KB_{S} and KB_{T} can be cast into two knowledge base graphs, G and G', as shown in Fig.1. The circles represent subjects or objects, and the triangles denote predicates. The predicates with the same name in different knowledge bases can be considered to be aligned.

④http://oaei.ontologymatching.org/, Nov. 2015.

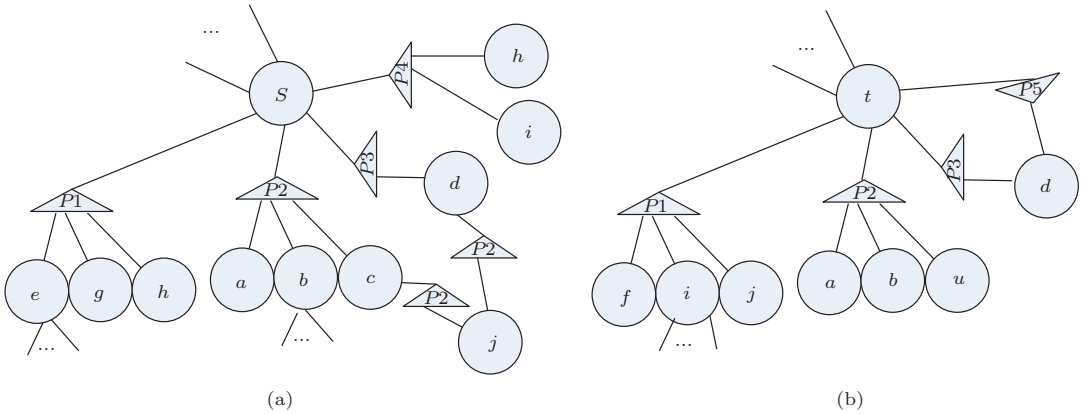


Fig.1. Graph representation of knowledge base. (a) Source knowledge base G. (b) Target knowledge base G'.

Instance Representation. An instance i can be represented as a set of facts corresponding to the same subject, i.e.,

$$R_i = \{(i, p, o) | p \in P,\ o \in I \cup L\}.$$

Given two knowledge bases to be aligned, if instance i aligns with j, then

$$i = \arg\max_{k \in I_{\mathrm{S}}} sim(k, j \mid G, G', \boldsymbol{B}_{-(k,j)}),$$

where $\boldsymbol{B} = \{B_{(i,j)}\}, i \in I_{\mathrm{S}}, j \in I_{\mathrm{T}}$ is the alignment matrix between I_{S} and I_{T}. When instances i and j are aligned, $B_{(i,j)} = 1$; otherwise, $B_{(i,j)} = 0$. $\boldsymbol{B}_{-(i,j)}$ is the alignment matrix $\boldsymbol{B}$ excluding the value of $B_{(i,j)}$. $sim(k, j \mid G, G', \boldsymbol{B}_{-(k,j)})$ is the similarity between k and j. We compute it based on the knowledge base graphs G and G' and the alignment matrix $\boldsymbol{B}_{-(i,j)}$, and we will discuss it in detail in Subsection 3.3. We now transform the problem of matching to the following optimization problem.

$$\max_{\boldsymbol{B}} \Sigma_{i \in I_{\mathrm{S}}, j \in I_{\mathrm{T}}} B_{(i,j)} \times sim(i, j \mid G, G', \boldsymbol{B}_{-(i,j)})$$
$$\text{s.t. } \Sigma_i B_{(i,j)} \leqslant 1, B_{(i,j)} \in \{0, 1\}.$$

By the reduction from the CLIQUE problem, one can show that this problem is NP-hard[9].

Observing the instances of knowledge bases, we find two characteristics of an instance.

Characteristic 1. An instance may have some information which most (even all) other instances do not have, which we call distinctive (unique) information.

Characteristic 2. An instance is related to some other instances. For example, an instance has some objects which are also instances and it can also be an object of some other instances.

Considering the two characteristics, the intuitive solutions for the above optimization problem are generating a set of candidate pairs using indexing with the distinctive information to avoid the similarity computation for all instance pairs and iteratively computing the similarities for the candidate pairs by incorporating new aligned instance pairs. The detail of our framework will be illustrated in Section 3. Before we go to the details, we first present two definitions which will be used in the next section according to the two characteristics.

Unique Instance Set. Unique instance set UI is a collection of instances containing tuples of $(predicate, object)$ which occur only once in a knowledge base, formally, $UI = \{i | \exists (i, p', o''), \forall j \neq i, (j, p', o'') \notin F\}$.

Compatible_Neighbors[7]. According to the second characteristic that an instance has relation with some other instances. If two instances are aligned, then the instances related to them may be aligned. We define the correlated instances as compatible_neighbors, CN. Formally,

$$CN(i, j) = \{(k, l) | (i, r, k) \in KB_{\mathrm{S}},\ (j, s, l) \in KB_{\mathrm{T}}\} \cup \{(k, l) | (k, r, i) \in KB_{\mathrm{S}},\ (l, s, j) \in KB_{\mathrm{T}}\},$$

where r and s are aligned predicates. For example, in Fig.1, $(e, f) \in CN(s, t)$, $(s, t) \in CN(e, f)$.

3 RiMOM-IM Framework

This section describes the detail of RiMOM-IM framework. The overview of the instance matching system is shown in Fig.2. The system includes five modules including initial interactive configuration, candidate pair generation, matching score calculation, instance alignment, and validation. The annotated num-

bers in the module are the sequential procedures. In the following, we illustrate its workflow and dataflow.

1) The system begins with initial interactive configuration, which allows users to input the data and configure the needed modules in the following process with their parameters.

2) We conduct data preprocessing, such as unifying data formats for the values of some predicates.

3) We proceed with the blocking which consists of using inverted indexing to generate candidate set and unique instance sets.

4) For each pair in the candidate set, we compute the similarities over all aligned predicates with "similarities over predicates" and then through "aggregation", we aggregate them to get the final matching score of two instances. By ordering the scores from high to low, we generate a priority queue.

5) For unique instance sets, we iteratively use "unique subject matching" and "one-left object matching" to generate aligned set until no new aligned instances are generated. These aligned instances will then be used to add new candidate pairs and find new unique instances, thus updating candidate set and unique instance sets. Correspondingly, the matching scores for related instance pairs and the priority queue will be updated.

6) For the priority queue, we use "score matching" to generate only one aligned pair with the highest score above the threshold. If there is a new aligned instance pair, we will generate new unique instances, which will be taken as the input in step 5. If there is no new aligned pair, we continue to step 7.

7) If "validation" module is chosen in step 1, we will conduct the validation on all aligned pairs; otherwise, we terminate. All the aligned instance pairs will be added into the aligned set.

The overview of the instance matching framework is summarized as follows:

- initial interactive configuration
- candidate pair generation
- matching score calculation
- do
  - do
    - ∗ unique subject matching
    - ∗ one-left object matching
  - until (no new matching pairs are generated)
  - score matching
- until (no new matching pairs are generated)
- validation
- return *AlignedSet*

We further illustrate each module in following subsections.

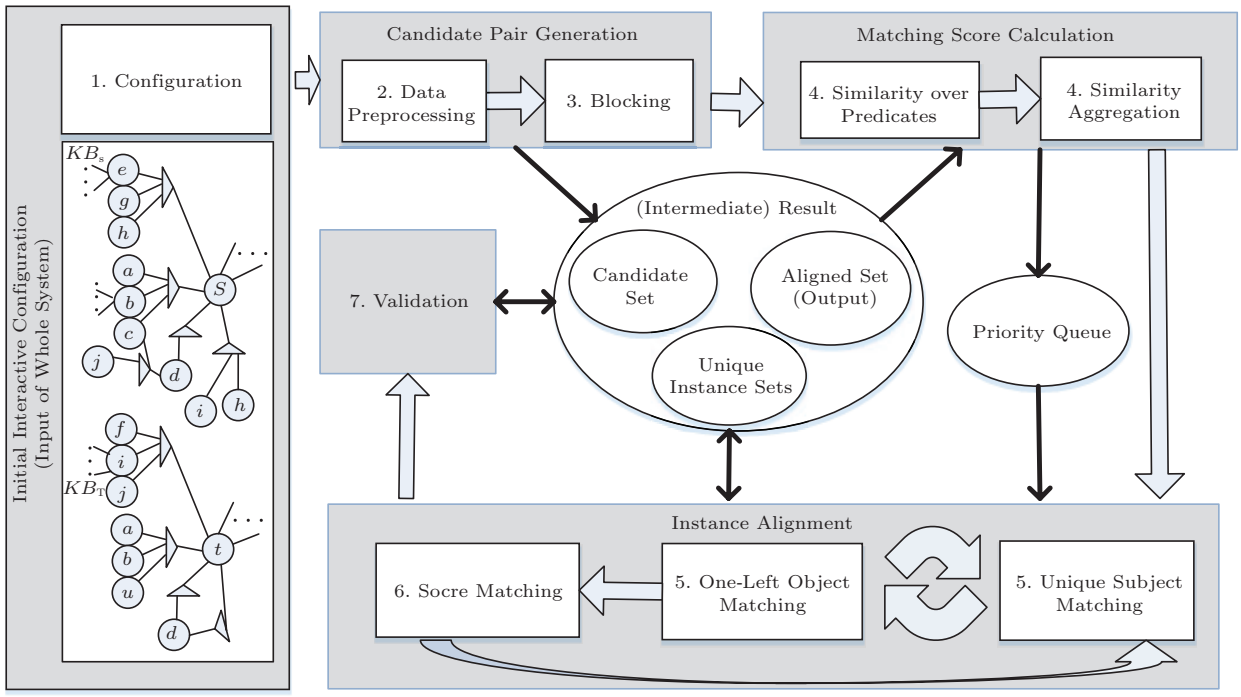


Fig.2. Framework of RiMOM-IM. We use boxes to represent processes (steps) and ellipses to represent data. The black narrow arrows are used to denote the data flow, while the grey wide arrows are used to indicate the process flow.

3.1 Initial Interactive Configuration

In terms of configuration, we can input the aligned predicates, and choose whether to use the Google translator for cross-lingual instance matching task. For each pair of aligned predicates, we select one or more predefined similarity functions. To aggregate the similarities of different predicates, we choose an aggregation function to get a final matching score. In this paper, we choose the proposed aggregation function, *ExpAgg*. The threshold used in "score matching" is also defined.

3.2 Candidate Pair Generation

Data Preprocessing. We remove special symbols like "♯", "*", "!" and stop words such as "a", "of", and "the". Afterwards, we calculate the TF-IDF values of words in each knowledge base. Note that the objects associated with a particular subject and predicate consist of a pseudo-document. In addition, since the data formats of different knowledge bases are inconsistent, we uniform some of the data formats. For example, some represent a date as "2014, 05, 01", some denote as "May 1st, 2014", and we unify them as "05-01-2014".

Blocking. Blocking aims to pick a relatively small set of candidate pairs from all pairs. Due to the large scale of knowledge bases, it is impossible to calculate the matching scores of all instance pairs. Most traditional methods use all the words except stop words in the vocabulary as the keys for indexing. For example, Diallo and Ba[10] used the words in the direct virtual document of an entity as the keys for indexing, where the direct virtual document of an entity is constituted by the combination of its uniform resources identifier, the URI, the local name, the labels in different languages extracted by a function and the set of annotations associated with it. Li *et al.*[11] built the inverted indexes for name vectors and virtual documents of an entity respectively. Nevertheless, they ignored the predicates and failed to choose the distinctive object features which are vital in pruning instance pairs for large-scale knowledge bases.

In our blocking method, we take the predicate as well as the top five words of the object (ordered by TF-IDF values in the knowledge base) as an index key to index instances. It should be noticed that if the object is an instance, the entire URI is considered as a word. Owing to the novel blocking method which restricts the candidate pairs with identical distinctive information (predicate and distinctive object features), we greatly reduce the similarity comparisons and improve the efficiency. The blocking can be divided into three phases. The detail of the algorithm is shown in Algorithm 1. The first phase (from line 2 to line 12) indexes every instance in each data source extracted from its RDF objects. The second phase (from line 13 to line 16) traverses the index table and generates candidate

Algorithm 1. Blocking

Input: $KB_{\mathrm{S}}, KB_{\mathrm{T}}$
Output: candidate set C, unique instances sets U_{S}, U_{T}
1 $C \leftarrow \emptyset$, $U \leftarrow \emptyset$, $A \leftarrow \emptyset$, $B \leftarrow \emptyset$
2 **foreach** $KB \in \{KB_{\mathrm{S}}, KB_{\mathrm{T}}\}$ **do**
3 **foreach** triple $(s, p, o) \in KB$ **do**
4 **if** not $A.ContainsKey(p + o)$ **then**
5 $A.addKey(p + o)$
6 $A.addValue(p + o, KB, s)$;
7 Split o into $words(o)$ and then sort the $words(o)$ by the TF-IDF values from high to low;
8 Extract 5 words from $words(o)$ with the top 5 TF-IDF values and appearing in the other knowledge base for at least one time.
9 **foreach** word $w_i \in$ top 5 TF-IDF words **do**
10 **if** not $B.ContainsKey(p + w_i)$ **then**
11 $B.addKey(p + w_i)$;
12 $B.add(p + w_i, KB, s)$;
13 **foreach** $key \in A.AllKeys()$ **do**
14 **foreach** *subject* $s_i \in A.getValues(key, KB_{\mathrm{S}})$ **do**
15 **foreach** $s_j \in A.getValues(key, KB_{\mathrm{T}})$ **do**
16 $C.add((s_i, s_j))$;
17 **foreach** $key \in B.AllKeys()$ **do**
18 **if** $B.getValues(key, KB_{\mathrm{S}}).size() == 1$ **then**
19 $U_{\mathrm{S}}.add(key, KB_{\mathrm{S}}, B.getValues(key, KB_{\mathrm{S}}))$;
20 **if** $B.getValues(key, KB_{\mathrm{T}}).size() == 1$ **then**
21 $U_{\mathrm{T}}.add(key, KB_{\mathrm{T}}, B.getValues(key, KB_{\mathrm{T}}))$;
22 Return $C, U_{\mathrm{S}}, U_{\mathrm{T}}$;

set C. The third phase (from line 17 to line 21) generates two unique instance sets U_{S} and U_{T}, one for KB_{S} and the other for KB_{T}, which will be used in instance alignment.

3.3 Matching Score Calculation

In this module, we aim to get the matching scores of instance pairs in candidate set. We obtain it by aggregating the similarities over aligned predicates.

Similarity over Predicates. We use different similarity functions for different aligned predicate pairs. Specifically, when the predicate pair's objects are a set of instances, we use Jaccard similarity to calculate the similarity between the instances of the predicate pair. When the predicate pair's objects are texts, we calculate the cosine similarity based on the bag-of-words representation of the texts as the similarity between the instances of the predicate pair. We implement a lot of these classical similarity functions as discussed in [12], thus users can specify the functions according to their specific cases in the system configuration step.

Similarity Aggregation. For each instance pair, after getting similarities over multiple aligned predicates, we need to aggregate the similarities to get the final matching score. AVG aggregates the similarities by computing the average value[13]. SIGMOID (SIG) aggregates the similarities by computing the average similarities transformed by a sigmoid function[13]. These methods do not adapt to the case when different instance pairs have different numbers of aligned predicates.

In this work, we propose a weighted exponential aggregation function, *ExpAgg*, to aggregate the similarities S, which is a set of similarities of all aligned predicates. The function is as follows:

$$ExpAgg(S) = \frac{\Sigma_{s_i \in S} w_i' \times \exp(w_i'' \times s_i)}{\Sigma_{s_i \in S} w_i' \times \exp(w_i'' \times 1)}.$$

Among them, s_i is the similarity over the i-th aligned predicate. The larger the value of s_i is, the more information the predicate carries. The parameter w' reflects the importance of a predicate, and w'' controls the importance of informative predicates. If w'' is too large, then the matching score will ignore the influence of much less informative predicates. If w'' is too small, much noise will be introduced by these predicates. In this paper, we set the parameters experimentally. Specifically, for each parameter, we vary the value from 1 to 10 with an interval 1, with the other parameter fixed. Finally, we use the best parameters $w_i' = 1$ and $w_i'' = 5$ that result in the best performance on our training data. As presented in Proposition 1, we show the proposed aggregation function puts more weight on more informative predicates, and it is more accurate than traditional aggregation methods. The experiment in Subsection 4.5 also indicates our aggregation method outperforms traditional methods. Furthermore, we prove it theoretically as following.

Proposition 1 (*ExpAgg* Favors More Informative Predicates). *When $s_i \geqslant \delta$, $s_j - s_i \geqslant \delta$, and $s_j \geqslant \Sigma_{i=1,\ldots,m} s_i, m < 100$, it is possible to choose an appropriate value k to let $e^{k \times s_j} > k \times \Sigma_{i=1,\ldots,m} e^{k \times s_i}$.*

Proof. There is a k that makes $e^{k \times (s_i - s_j)} \leqslant e^{-\delta \times k} < 1/100 \times k$, thus, $\Sigma_{i=1,\ldots,m} e^{k \times s_i - s_j} < 1/k$, that is, $e^{k \times s_j} > k \times \Sigma_{i=1,\ldots,m} e^{k \times s_i}$. Therefore *ExpAgg* favors more informative predicates. □

Aggregation function favoring more informative predicates will tend to get the right matching results. Suppose we have three instances a, b and c, if $sim(a,b) = (1,1,0)$, $sim(a,c) = (1,0.7,0.5)$, after the aggregation with AVG[13], $AVG(sim(a,b)) = 0.667$, $AVG(sim(a,c)) = 0.733$; while with the method of SIG[13], $SIG(sim((a,b)) = 0.661$, $SIG(sim(a,c)) = 0.675\,5$; with our proposed method of *ExpAgg*, $ExpAgg(sim(a,b)) = 0.668\,9$, $ExpAgg(sim(a,c)) = 0.435\,1$. Then AVG and SIG will choose (a,c) as the aligned one, while our aggregation function will choose (a,b) as aligned, which is as expected. When $sim(a,c) = (1,0.9,0.9)$, our method will get $ExpAgg(sim(a,c)) = 0.737\,7$, and our aggregation will choose (a,c) to be aligned as expected. Therefore, our aggregation function tends to believe the predicate which carries more information. Since the aligned pairs always have some precise information, our aggregation function can work well when the numbers of aligned predicates of different instance pairs are unbalanced.

3.4 Instance Alignment: Three Strategies

Most traditional methods of instance matching will determine the alignment of instance pairs based on the matching scores. However, since a matching score of an instance pair will be influenced by the alignment situation of the compatible_neighbors of the two instances of the pair and we do not know the alignment situation initially, the matching score is likely to be unreliable.

In this work, we propose three different strategies for instance matching. We use "unique subject matching" and "one-left object matching" to determine some aligned pairs at the very beginning. Afterwards, based on the aligned instances, we update related matching

scores. By ordering matching scores from high to low, we extract one pair with the highest score above the threshold as one aligned pair, which is called "score matching". The three strategies work in an iterative way.

Unique Subject Matching. Owing to the large scale of knowledge bases, if an indexing key is contained in both unique instance sets U_{S} and U_{T}, then the corresponding instance pair will be aligned. Therefore, we can avoid aligning instances via their matching scores when the scores are incorrect initially. As the aligned pairs will generate new aligned pairs (for example, if we know (a, b) is aligned now, and $c \in U_{\mathrm{S}}$ is indexed with $p+a$, $d \in U_{\mathrm{T}}$ is indexed with $p+b$, then (c, d) is aligned now), we iteratively generate new aligned pairs based on the already aligned pairs until no new aligned pairs are generated. The aligned pairs will be used to update candidate set and matching scores correspondingly.

"Unique subject matching" can align instances via unique instance set which dose not need much overlapping information. Thus if two knowledge bases have little overlapping information, our system can still work well.

One-Left Object Matching. We use characteristic 2 of an instance to generate aligned pairs. In detail, if two aligned instances have the same predicate with m objects, of which $m-1$ are aligned, then the "one-left" objects are aligned. For example, in the knowledge base graph as shown in Fig.1, if subject (t, s) is aligned, considering the predicate $P2$, the left one object pair (u, c) can be concluded as aligned. Similarly, we iteratively use the aligned object pair to generate more aligned objects until no new object pairs are aligned. Then we use the aligned object pairs to update the unique instance sets, the candidate set and the matching scores correspondingly.

One-left object matching relies on aligned instance pairs and their compatible_neighbors, thus if a knowledge base has little overlapping information but has rich information of correlations among instances, our system can work as well.

Score Matching. Since previous processes have generated some reliable aligned pairs which can be utilized for the similarity score computation of instance pairs, we can consider the instance pair with the highest matching score above a predefined threshold δ as aligned. Note that to guarantee accuracy, each time we get only one aligned instance pair with the highest score. Then we add its compatible_neighbors to candidate set, and update unique instance sets and related matching scores correspondingly.

With the greedy algorithm of extracting only the most matching pair every time, we control the error propagation to some extent. As we cannot guarantee a global optimization with the greedy algorithm, we add the process of validation.

3.5 Validation

Since many objects of instances are URIs referring to other instances, there still exists some nondeterminacy in aligning two instances due to the uncertainty in the alignment situation of their compatible_neighbors. We add validation module to correct some mistakes by recomputing the final matching scores of aligned instance pairs. If the recomputed score of a pair is lower than the threshold δ, we remove it. In our experiment, we set $\delta = e^{0.5}$.

4 Experiment

In this section, we present experimental results to demonstrate the effectiveness of the proposed approach. The source code of our proposed framework, RiMOM-IM in the paper, is publicly available[5].

4.1 Datasets and Experimental Setup

We evaluate our approach using the OAEI@IM datasets, and compare our system with the other OAEI participants'. OAEI[6] is an annual ontology matching competition that provides authoritative tests and evaluations of ontology matching technologies. All datasets can be downloaded from the corresponding OAEI webpages. The dataset statistics are given in Table 1.

Table 1. Datasets Statistics

Dataset	Number of Concepts	Number of Properties	Number of Instances	Cross-Lingual
IM@2014	4	6	1 330	Yes
IM@2013	0	11	430	Yes
IIMB@2011	29	22	12 333	No
IIMB@2010	29	22	1 416	No

Competition Datasets. The dataset of IM@2014 contains two sub-datasets, namely, the id-rec dataset

[5] http://keg.cs.tsinghua.edu.cn/project/RiMOM/, Nov. 2015.

[6] http://http://oaei.ontologymatching.org, Nov. 2015.

and the sim-rec dataset. The datasets contain the instances describing famous books with different genres and topics. The id-rec dataset is a typical evaluation task of instance matching tools where the goal is to determine when two OWL instances describe the same real-world entity. The sim-rec task focuses on the evaluation of the similarity degree between two OWL instances, even when the two instances describe different real-world entities. The source Abox contains 173 book instances and the target Abox contains 172 book instances. Then we need to provide $173 \times 172 = 29\,756$ mappings, each one featured by a degree of similarity in the range [0, 1]. Given a mapping m, we need compare the similarity degree assigned to m by the matching tool against the corresponding similarity degree assigned by more than 250 workers[14]. The evaluation will be performed through the Euclidean distance, which means the smaller, the better. The datasets of IM@2013 are called RDFT test cases. The RDFT test cases have been generated from an initial RDF dataset, which is about well-known computer scientists extracted from DBpedia. Starting from the initial dataset, different transformations have been implemented. OAEI organizer provides the participants with five test cases. The RDFT test cases provide transformation techniques supporting value transformation, structure transformation, language transformation, and cardinality transformation.

Large-Scale Datasets. We choose the IIMB datasets of IM@2011 and IM@2010. We have also considered IM@2012, but it is not available for downloading now. Both the 2011 and the 2010 edition of IIMB are created by extracting data from Freebase, and each dataset consists of 29 concepts, 22 properties and thousands of instances divided into 80 test cases. The numbers of instances of datasets IM@2010 and IM@2011 are 1 416 and 12 333 respectively. For the IIMB@2011 dataset, it is not suitable to calculate the similarities of every possible instance correspondence any more because this will result in approximative $12\,333 \times 12\,333 = 152\,102\,889$ similarity comparisons. Thus, IIMB@2011 dataset is large enough to be used for large-scale alignment task.

Same as previous studies, to evaluate the blocking step, we use pair reduction ratio (RR) and pair completeness (PC). For interlinking results, we use presicion (Pre), recall (Rec), and $F1$-measure ($F1$). Since the last three measures are well-known, we only give formulas of RR and PC:

$$RR = 1 - \frac{\text{number of candidates}}{\text{number of all pairs}},$$

$$PC = \frac{\text{number of correct candidates}}{\text{number of actually aligned pairs}}.$$

Over these benchmark datasets, we test our framework on the tasks of both competition datasets and large-scale knowledge base alignment. Afterwards, we evaluate the performance of proposed blocking for candidate pair generation. Finally, we present a system analysis to evaluate our proposed novel strategies for direct instance matching and similarity aggregation method.

4.2 Experiments on Competition Datasets

In this subsection, we test the performance of RiMOM-IM on the dataset of IM@2014 and IM@2013 and compare it with other systems[15] (see Fig.3, Fig.4 and Table 2 for details). We can see that RiMOM-IM significantly outperforms the other methods on all datasets, especially on the id-rec dataset and the sim-rec dataset of IM@2014. On the id-rec dataset, we achieve the $F1$ value of 0.558 1, and the runner-up system gets only 0.099 1. On the sim-rec dataset, the cardinality of the reference alignment is 4 104 mappings. In the analysis, we are interested in comparing the similarity degree σ of mapping m_c against the similarity degree calculated by the matching tools of participates'. The result of this mapping analysis shows that RiMOM-IM has a similarity degree close to the expected value.

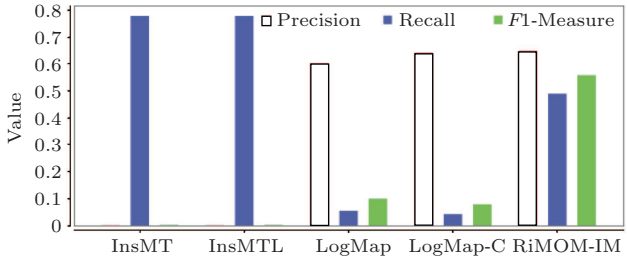


Fig.3. Identity relognition task results for the id-rec dataset of IM@2014.

We can see RiMOM-IM performs the best on almost all the five cases of IM@2013, with an average $F1$ value of 0.974 (+18% significant improvement compared with LilyIOM and +3.6% improvement compared with SLINT+). On the first two cases, which do not contain any cross-lingual properties, the performance is equal to SLINT+. While on test cases 3, 4 and 5, which contain cross-lingual properties, we use Google translator to bridge the multi-lingual gap, our system is much better with respect to both precision (Pre) and recall (Rec). In addition, the $F1$ values of our system in all

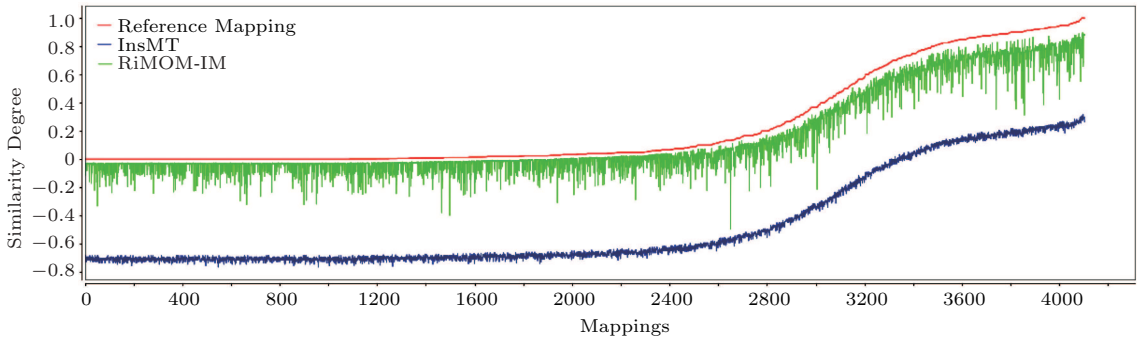


Fig.4. Similarity recognition task results for the sim-rec dataset of IM@2014.

Table 2. Results for IM@2013

System	Test Case 01			Test Case 02			Test Case 03			Test Case 04			Test Case 05		
	Pre	Rec	$F1$	Pre	Rec	$F1$	Pre	Rec	$F1$	Pre	Rec	$F1$	Pre	Rec	$F1$
LilyIOM	1.00	0.99	1.00	0.74	0.74	0.74	0.94	0.92	0.93	0.71	0.73	0.72	0.71	0.49	0.58
LogMap	0.97	0.69	0.80	0.79	0.99	0.88	0.98	0.73	0.84	0.95	0.70	0.80	0.92	0.62	0.74
SLINT+	0.98	0.98	0.98	**1.00**	**1.00**	**1.00**	0.94	0.91	0.92	0.91	0.91	0.91	0.87	0.88	0.88
RiMOM-IM	**1.00**	**1.00**	**1.00**	0.95	0.99	0.97	0.96	**0.99**	**0.98**	**0.94**	**0.98**	**0.96**	**0.93**	**0.99**	**0.96**

datasets are all greater than 0.96. As for the relatively poor performance of other systems on the last three test cases, as mentioned by the OAEI organizers[15], working in the direction of improving the combination and balancing of different matching techniques in a single, general-purpose, configuration scheme is a possible challenge for instance matching tools. We think that the reason why our system behaves much better is probably because we propose the aggregation function, *ExpAgg*, which addresses the imbalance of aligned predicate numbers among different instance pairs.

4.3 Experiments on Large-Scale Datasets

In this subsection, we test the performance of RiMOM-IM on the datasets of IM@2011[16] and IM@2010[17] and compare it with other systems. In the dataset of IIMB of IM@2011, many alignment results cannot be found even by human. We have asked the manager of the dataset about the problem. Unfortunately, we are told that the cooperators of the dataset have left the team, thus they cannot tell the reason either. As a result, we only list the data results that can be validated by human, namely, the test cases of 1∼20 and 41∼60, and we also give the results of all the datasets in IM@2010 (see Fig.5). We can see from Fig.5, on the dataset of IIMB of IM@2010, our system outperforms all the other systems under all measures especially in recall. It attributes to our strategy of candidate pair selection. Our system can pick a relatively small set of candidate pairs in order to avoid the

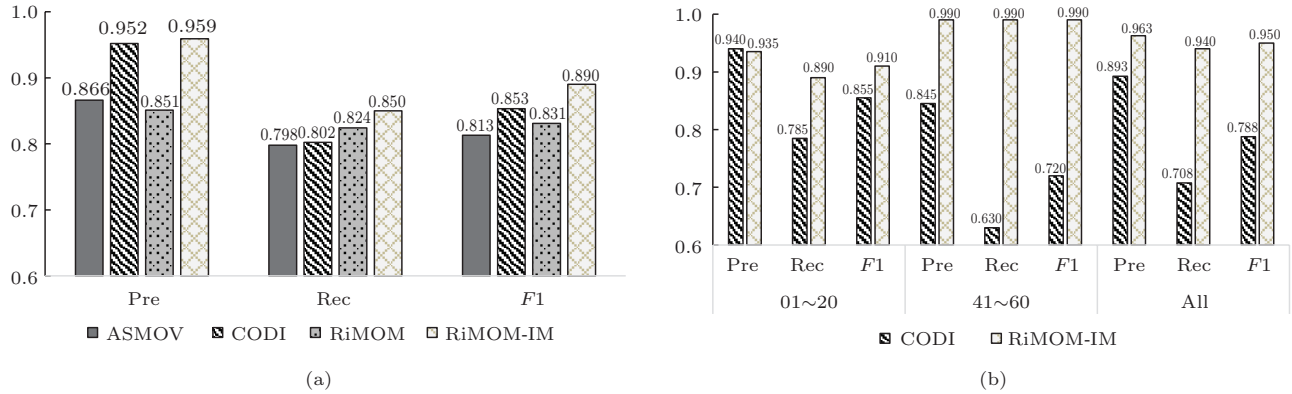


Fig.5. Results of (a) IM@2010 and (b) IM@2011.

similarity computation for all pairs, while it ensures a high pair completeness and avoids losing many instance pairs which may be aligned. Compared with existing RiMOM system[8], our RiMOM-IM achieves significant improvement in terms of both $F1$ measure and time efficiency (more than 200 times faster)

On datasets of 2011, our system behaves significantly better than the only successful participant, CODI, both in precision and recall, with $F1$ value of each test case exceeding 0.9. On the 41∼60 test cases, our $F1$ value reaches almost 1.0, while CODI's value is only 0.72. On test cases 01∼20, CODI only achieves similar precision as our system, while their $F1$ value is much less than ours. On all the above test cases, our average $F1$ value reaches 0.95, which means our system can be put into practical usage. All the 40 test cases contain about 6 million triples, and the execution time of RiMOM-IM when running on a desktop machine equipped with 3.30 Ghz dual-core CPU and 4 GB of memory is 29.1 minutes, which proves that our framework is fast and efficient on large-scale datasets.

4.4 Evaluation of Candidate Pair Generation

In our system, we generate initial candidate instance pairs via blocking. Each time when we find a new matching pair, we will add their compatible_neighbors into the candidate set. We use pair reduction ratio (RR), pair completeness (PC) and recall to evaluate the candidate pair generation, as shown in Table 3.

Table 3. Evaluation of Candidate Pair Generation

Dataset	# Instance Pairs	# Candidate Pairs	RR	PC	Recall
IM@2013	924 500	25 605	0.970	0.92	0.99
IM@2011	629 106 720	1 462 804	0.998	0.91	0.94
IM@2010	190 221 120	387 840	0.998	0.82	0.85

Note: # means "Number of".

Observing Table 3, we can easily find out why our framewrok is so efficient. As we can see from the latter two rows, on large-scale datasets, our RR values all exceed 0.998. On the dataset of IM@2010, the number of instance pairs is 490 times greater than that of our selected candidate pairs. On the dataset of IM@2011, it is about 430 times. By reducing so many instance pairs, there is no doubt that our framework works efficiently. Specially, on a personal computer with 4 GB memory and 3.3 GHZ frequency, the datasets of IM@2010, IM@2011 and IM@2013 only take 5.3 minutes, 29.1 minutes and 9 seconds respectively.

From the values of recall and pair completeness (PC) of the table, we may be confused why recall value can be greater than PC values. It is because every time when we find aligned pairs, we will add compatible_neighbors to the collection of candidate pairs. Thus, the instance pairs that have not been found in the blocking step can enter into the candidate set. The discovery of aligned pairs would not be limited to the candidate pairs found by the blocking procedure.

In summary, without blocking procedure, we cannot efficiently find the candidate pairs, while without compatible_neighbors as complement, we may leave out many aligned pairs. With both steps, we successfully generate candidate pairs with high RR, PC and recall.

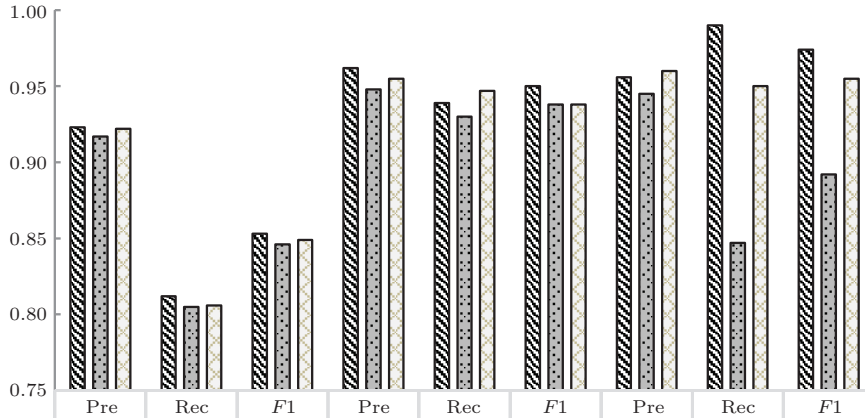
4.5 System Analysis

In this experiment, we explore the influence of our new strategies which directly align instances and the proposed aggregation method compared with simple average aggregation (AVG). The results are presented in Fig.6. "All+*ExpAgg*" means that we use all the modules mentioned in this paper with our proposed *ExpAgg* as similarity aggregation function. "All-UO" represents the system without the matching modules of "unique subject matching" and "one-left object matching", using *ExpAgg*. "All+AVG" denotes using average aggregation instead of the proposed *ExpAgg*.

Fig.6 indicates that using all modules achieves the best result. The baseline method "All-UO" has the lowest performance due to the removal of the modules of "unique subject matching" and "one-left object matching". The reason is that the aligned pairs generated by these two strategies are highly reliable and cannot be acquired by traditional methods of similarity calculation. Through the comparison between the results of "All+*ExAgg*" and "All+AVG", we can see that our method "all" has a significant improvement on the dataset of IM@2013. By looking into the datasets, we find that the reason is due to the serious imbalance of predicates in the knowledge base of IM@2013. Therefore, our aggregation method, *ExpAgg*, which is designed to address the problem of imbalance, achieves outstanding performance.

5 Related Work

Recently, a lot of inspiring knowledge base matching algorithms and tools have been proposed. From



	Pre	Rec	*F*1	Pre	Rec	*F*1	Pre	Rec	*F*1
	2010			2011			2013		
All+*ExpAgg*	0.923	0.812	0.853	0.962	0.939	0.950	0.956	0.990	0.974
All-UO	0.917	0.805	0.846	0.948	0.930	0.938	0.945	0.847	0.892
All+AVG	0.922	0.806	0.849	0.955	0.947	0.938	0.960	0.950	0.955

Fig.6. Module analysis.

the view of purposes, some of them focus on schema matching[13,18], some focus on instance matching[7,19] and the others can solve both schema and instance matching problems[4,6,8]. In terms of methods, there are heuristic methods[20-21], probabilistic methods[6,22], graph-based methods[23], learning-based methods[24-25] and reasoning-based methods[4,13,26]. A comprehensive survey is given by [1]. In this paper, due to the limitation of space, we only describe some closely related systems and their methods. The first work is reasoning-based and the others are all similarity-based.

Combinatorial optimization for data integration (CODI)[4] uses the terminological structure for knowledge base matching. It is based on the syntax and semantics of Markov logic and transforms the alignment problem to an optimization problem. Therefore, it relies on the schema of knowledge base and may not work efficiently when dealing with knowledge bases with little schema information. ASMOV[13] is an automated ontology matching system that uses a weighted sum to combine a comprehensive measure of similarity with a validation technique. ASMOV computes four types of similarities, including a lexical similarity, two structural similarities and an external similarity. Nevertheless, when ontologies have no information of property types, their similarity measures may perform poor. SLINT[19] is an instance matching system that does not need manually aligned predicates. When computing the similarities between instances, it first computes the similarities between all predicates and then incorporates the predicate similarities into instance similarities. SLINT may perform well when aligning knowledge bases containing many predicates, but its performance will be influenced by the noise of predicates when predicates can be easily aligned manually. SiGMa[7] is an iterative propagation algorithm which leverages both the structural information from the relationship graph and the flexible similarity measures between entity properties in a greedy local search. As we also use the correlation between instances and have different similarity measures for different aligned predicates, our approach is close to SiGMa.

Overall, all the related approaches except SiGMa do not utilize the iterative framework and fail to make the best use of aligned instances. Though all the methods use the blocking technique to select candidates, only our blocking method takes both predicates and distinctive object features as indexing keys, which is more efficient. All the above approaches do not consider the case when the number of aligned predicates between instances varis largely, which results in the loss of the accuracy of similarities between instances. We propose a weighted exponential similarity aggregation method to successfully address it. Finally, we propose the strategies for direct instance matching without similarity computation and it can advance similarity-based strategy, especially at the beginning (cold start).

6 Conclusions

In this paper, we investigated instance matching which is a critical problem in knowledge sharing and

integration. We presented an iterative matching framework RiMOM-IM which achieves substantial improvement compared with competitive state-of-the-art instance matching systems. In the proposed framework, we proposed a distinctive information based blocking method to greatly reduce the number of candidate instance pairs, which greatly improves efficiency and adapts our framework to large-scale datasets. We also presented a weighted exponential function based similarity aggregation function, which addresses the problem of unbalanced aligned predicate numbers among different instance pairs. In addition, two strategies, "unique subject matching" and "one-left object matching" for direct instance matching without computing the similarities, were put forward to advance similarity-based alignment module, which ensures a high reliability of the similarities. With the iterative matching scheme and an additional validation process, we made best use of existing aligned instance pairs and achieved an impressing performance.

In future work, first, we will improve our work by configuring the parameters for different tasks automatically instead of manually. Second, as in this work, we assumed the predicates which can be aligned have already been aligned, we will incorporate the process of aligning predicates into our framework to make our system more powerful.

References

[1] Shvaiko P, Euzenat J. Ontology matching: State of the art and future challenges. *IEEE Trans. Knowl. Data Eng.*, 2013, 25(1): 158-176.

[2] Ferrara A, Nikolov A, Noessner J *et al.* Evaluation of instance matching tools: The experience of OAEI. *Web Smantics: Science, Services and Agents on the World Wide Web*, 2013, 21: 49-60.

[3] Bellahsene Z, Bonifati A, Rahm E. Schema Matching and Mapping. Springer-Verlag Berlin, Heidelberg, 2011.

[4] Huber J, Sztyler T, Noessner J *et al.* CODI: Combinatorial optimization for data integration—Results for OAEI 2011. In *Proc. the 6th International Workshop on Ontology Matching*, Oct. 2011, pp.134-141.

[5] Volz J, Bizer C, Gaedke M, Kobilarov G. Discovering and maintaining links on the web data. In *Proc. the 8th International Semantic Web Conference*, Oct. 2009, pp.650-665.

[6] Suchanek F M, Abiteboul S, Senellart P. PARIS: Probabilistic alignment of relations, instances, and schema. *PVLDB*, 2011, 5(3): 157-168

[7] Lacoste-Julien S, Palla K, Davies A, Kasneci G, Graepel T, Ghahramani Z. SIGMa: Simple greedy matching for aligning large knowledge bases. In *Proc. the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2013, pp.572-580.

[8] Li J, Tang J, Li Y, Luo Q. RiMOM: A dynamic multistrategy ontology alignment framework. *IEEE Trans. Knowl. Data Eng.*, 2009, 21(8): 1218-1232.

[9] Böhm C, de Melo G, Naumann F, Weikum G. LINDA: Distributed web-of-data-scale entity matching. In *Proc. the 21st CIKM*, Oct.29-Nov.2, 2012, pp.2104-2108.

[10] Diallo G, Ba M. Effective method for large scale ontology matching. In *Proc. the 5th SWAT4LS*, Nov. 2012.

[11] Li J, Wang Z, Zhang X *et al.* Large scale instance matching via multiple indexes and candidate selection. *Knowledge-Based Systems*, 2013, 50: 112-120.

[12] Euzenat J, Valtchev P. Similarity-based ontology alignment in OWL-lite. In *Proc. the 16th ECAI*, August 2004, pp.333-337.

[13] Jean-Mary Y R, Shironoshita E P, Kabuka M R. Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2009, 7(3): 235-251.

[14] Dragisic Z, Eckert K, Euzenat J *et al.* Results of the ontology alignment evaluation initiative 2014. In *Proc. the 9th International Workshop on Ontology Matching*, Oct. 2014, pp.61-104.

[15] Grau B C, Dragisic Z, Eckert K *et al.* Results of the ontology alignment evaluation initiative 2013. In *Proc. the 8th International Workshop on Ontology Matching*, Oct. 2013, pp.61-100.

[16] Euzenat J, Ferrara A, van Hage W R *et al.* Results of the ontology alignment evaluation initiative 2011. In *Proc. the 6th Internaitonal Workshop on Ontology Matching*, Oct. 2011.

[17] Euzenat J, Ferrara A, Meilicke C *et al.* Results of the ontology alignment evaluation initiative 2010. In *Proc. the 5th International Workshop on Ontology Matching*, Nov. 2010.

[18] Do H H, Rahm E. COMA: A system for flexible combination of schema matching approaches. In *Proc. the 28th International Conference on Very Large Data Bases*, Aug. 2002, pp.610-621.

[19] Nguyen K, Ichise R, Le B. SLINT: A schema-independent linked data interlinking system. In *Proc. the 7th Internaitonal Workshop on Ontology Matching*, Nov. 2012.

[20] Hu W, Qu Y. Falcon-AO: A practical ontology matching system. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2008, 6(3): 237-239

[21] Pirrò G, Talia D. UFOme: An ontology mapping system with strategy prediction capabilities. *Data Knowl. Eng.*, 2010, 69(5): 444-471.

[22] Albagli S, Ben-Eliyahu-Zohary R, Shimony S E. Markov network based ontology matching. In*Proc. the 21st IJCAI*, Jul. 2009, pp.1884-1889.

[23] Melnik S, Garcia-Molina H, Rahm E. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proc. the 18th ICDE*, Feb.26-Mar.1, 2002, pp.117-128.

[24] Ehrig M, Staab S, Sure Y. Bootstrapping ontology alignment methods with APFEL. In *Proc. the 18th WWW* (*Special Interest Tracks and Posters*), May 2005, pp.1148-1149.

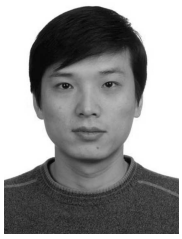
[25] Doan A, Madhavan J, Dhamankar R, Domingos P, Halevy A Y. Learning to match ontologies on the semantic web. *VLDB J*, 2003, 12(4): 303-319.

[26] Niepert M, Meilicke C, Stuckenschmidt H. A probabilistic-logical framework for ontology matching. In *Proc. the 24th AAAI*, Jul. 2010.

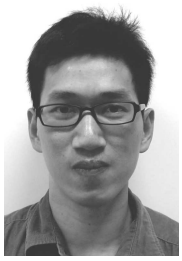
Chao Shao is a master student in the Department of Computer Science and Technology, Tsinghua University, Beijing. His research interests include instance matching, semantic web, text mining, and big data.

Lin-Mei Hu is a Ph.D. student in the Department of Computer Science and Technology, Tsinghua University, Beijing. Her research interests include text mining, topic models and semantic web.

Juan-Zi Li is a professor in the Department of Computer Science and Technology, Tsinghua University, Beijing. She received her Ph.D. degree in computer science and technology from Tsinghua University, Beijing, in 2000. Her research interests include semantic web and knowledge graph, news and social network mining.



Zhi-Chun Wang is an associate professor in the College of Information Science and Technology, Beijing Normal University. He received his Ph.D. degree in information management and system in Tianjin University, Tianjin, in 2010. His main research interest is knowledge graph building and mining.



Tonglee Chung is a Ph.D. candidate in the Department of Computer Science and Technology, Tsinghua University, Beijing. His research interests include knowledge base construction and machine learning.

Jun-Bo Xia is a master student in the Department of Computer Science and Technology, Tsinghua University, Beijing. His research interests include news mining and semantic web.