IEICE TRANSACTIONS on Information and Systems

VOL. E99-D NO. 7
JULY 2016

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

PAPER

A Heuristic Expansion Framework for Mapping Instances to Linked Open Data

Natthawut KERTKEIDKACHORN[†,††a)], ***Nonmember and*** **Ryutaro ICHISE**[†,††], ***Senior Member***

SUMMARY Mapping instances to the Linked Open Data (LOD) cloud plays an important role for enriching information of instances, since the LOD cloud contains abundant amounts of interlinked instances describing the instances. Consequently, many techniques have been introduced for mapping instances to a LOD data set; however, most of them merely focus on tackling with the problem of heterogeneity. Unfortunately, the problem of the large number of LOD data sets has yet to be addressed. Owing to the number of LOD data sets, mapping an instance to a LOD data set is not sufficient because an identical instance might not exist in that data set. In this article, we therefore introduce a heuristic expansion based framework for mapping instances to LOD data sets. The key idea of the framework is to gradually expand the search space from one data set to another data set in order to discover identical instances. In experiments, the framework could successfully map instances to the LOD data sets by increasing the coverage to 90.36%. Experimental results also indicate that the heuristic function in the framework could efficiently limit the expansion space to a reasonable space. Based upon the limited expansion space, the framework could effectively reduce the number of candidate pairs to 9.73% of the baseline without affecting any performances.

key words: *semantic web, linked data, Linked Open Data set, expansion space, search space, heuristic function, instance matching*

1. Introduction

In the big data era, interlinking among data sets is the key procedure to utilize information more wisely. For example, if an instance in data set A connects to the identical instance in data set B, the information of the instance in data set A could be enriched by its connected instance. As a result, another perspective of knowledge will be discovered. Therefore, Linked Data [1] was created to provide a simply concept of publishing and connecting such data. The aim of Linked Data is to construct a Linked Data collection or the web of Data. For building Linked Data, Linked Data concept specifies that the published data must be published under the Resource Description Framework (RDF) [2], must represent things by Uniform Resource Identifier (URI) and must be published in triples (subject, predicate, object). Currently, there is the ongoing project, which aims to construct Linked Data data sets, named the Linked Open Data (LOD) cloud [3]. In the LOD cloud, there are more than million instances. Consequently, it is well-known that any instances mapped to the LOD could be enriched by other instances in the LOD cloud. Therefore, the aim of this research is to discover and map instances to their identical instances in the LOD cloud.

Mapping instances to the LOD cloud becomes a challenging problem because of the continuous growth of LOD data sets. When the LOD project cloud started in 2007, there are only 12 data sets, while currently there are more than 1000 data sets [4]. Due to a large number of LOD data sets, we could not know which data set contains an identical instance. As a result, some source instances could not be mapped to the LOD cloud. To the best of our knowledge, mapping instances to more than a data set is not addressed yet.

In this article, we introduce HMILDs: a Heuristic expansion framework for Mapping Instances to LoD data sets (HMILDs). The basic idea of HMILDs is to directly map instances to one particular data set and then gradually expand a search space for discovering identical instances to other LOD data sets in order to find other identical instances. Due to a large amounts of instances in LOD data sets, an expansion strategy and a heuristic function for limiting the expanding search space are designed into the framework.

The rest of this article is organized as follows. Firstly, technical terms and a definition of mapping instances to LOD data sets are given in Sect. 2. Secondly, related works are briefly reviewed and are discussed in Sect. 3. Thirdly, the methodology of HMILDs is described in Sect. 4. In Sect. 5, the details of experiments and their results are presented. Eventually, this article is concluded in the last section.

2. Preliminary

To clarify the problem for mapping instances to the LOD cloud and some technical terms in the article, their definitions are given in this section.

An instance represents a real-world thing. Let a, e, s, x and y are instances.

Problem Mapping instances to the LOD cloud: Mapping instances to the LOD cloud is not similar to conventional mapping instances, which aims to map instances to a data set. Since the LOD contains tremendous amounts of data sets, mapping instances to the LOD cloud aims to map instances to more than a data set or multiple LOD data sets. Given $D_1, D_2, D_3, \ldots, D_n$ and S, where D_i represents the LOD data set i, n is a

Manuscript received September 24, 2015.
Manuscript revised January 26, 2016.
Manuscript publicized April 5, 2016.

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

DOI: 10.1587/transinf.2015EDP7390

number of LOD data sets and a source data set S, the definition of mapping instances to the LOD cloud is to compute set $\omega = \{(x, y) \,|\, x \equiv y, x \in S, \exists i\, (y \in D_i)\}$.

Definition 1 Identical Instance: $x \equiv y$ denoted that x and y are identical if and only if they are referred to the same real-world object.

Definition 2 Source Instance: s is an source instance if $s \in S$, where S is a source data set. A source data set is a data set that aims to map to the LOD cloud.

Definition 3 Anchor Instance: a is an anchor instance if $a \in A$, where A is an anchor data set. An anchor data set is a base data set in the LOD cloud where a source data set is mapped to at first. Any LOD data set could be selected as an anchor data set. However, a data set strongly connected to other LOD data sets is preferred. If the anchor data set tightly connect to other LOD data sets, it will provide useful links to expand the search space to another data set.

Definition 4 Adjacent Instance: y is the adjacent instance of x, when the following set is not empty:

$$\{(o_x, s_y) \,|\, < s_x, p_x, o_x > \in t(x), \\ < s_y, p_y, o_y > \in t(y) \text{ and } o_x \equiv s_y\}$$

, where $t(x)$ and $t(y)$ are a set of triples of instance x and y respectively and $< s_x, p_x, o_x >$ is a triple of $t(x)$ and $< s_y, p_y, o_y >$ is a triple of $t(y)$.

Definition 5 Expanded Instance: e is an expanded instance, if $e \in E$, where E is an expanded data set. An expanded data set is a LOD data set that could be reached from the anchor data set.

3. Related Works

Instance mapping, also known as instance matching, object co-reference resolution, or entity resolution, is the problem, which aims to discover two identical instances in the same data set or between difference data sets.

There are many approaches [5]–[12] proposed for mapping instances to a data set. In Silk [5], three steps are introduced in order to discover and manipulate the matching between different data sets. For the first step, a discovery engine computes identical links between different data sets. Then, the second step fine-tunes the correctness of such links. The third step manipulates the links when changing of data sets is applied. AgreementMaker [6] is a resolution system for matching both ontologies and instances. In AgreementMaker, three phases are performed in order to match between instances. Firstly, candidate pairs of instances are selected by similarity between labels of instances in the candidate generation phase. Secondly, similarity between instance pairs are extracted during the disambiguation phase. Finally, instance pairs are verified, whether they are correct match or not, in the matching phase. In Zhishi.Links [7], which is an enhanced version of Silk, some weighting schema is applied to improve the matching results between instance pairs. ObjectCoref [8] is a self-learning system, which detects identical objects by iteratively learning discriminative property. SERIMI [9] selectes high entropy predicates, which usually possess abilities to discriminate identical objects, in order to select instance pairs and then build a binary classifier to classify whether such instance pairs are correctly matched or not. SLINT [10] and SLINT+ [11] select useful predicates for generating candidate pairs of instances and then such candidate pairs are verified whether they are identical or not. Rong et al. later introduce an instance matching approach using similarity metrics [12]. Several types of the similarity metric are proposed to extract similarity features between candidate instances and then a binary classifier is employed to justify whether candidate pairs are matched. Although those approaches success to identify identical instances, such approaches could not perform mapping instances to multiple data sets.

Owing to a large amount of LOD data sets, we could not know which data set contains an identical instance. As a result, an instance matching system, which map instances to a data set, cannot effectively map instances to the LOD cloud. Kertkeidkachorn et al. therefore introduce an automatic instance expansion framework for mapping instances to the LOD [13]. The framework discovers and maps instances to the LOD cloud by gradually expanding the data set from one data set to another data set. Although their work successfully increases coverage of mapping instances to the LOD cloud, the search space during the expansion process from one data set to another data set is still high. In this article, the major contribution differentiating from the work [13] is a heuristic function for the automatic instance expansion framework in order to limit the expanded search space in the LOD cloud to a reasonable range. Moreover,we also give rigid evidences why mapping instances to multiple data sets are necessary for the LOD cloud and also provide further empirical study of similarity metrics in the instance matching component.

4. HMILDs

In this section, the details of HMILDs are presented. As depicted in Fig. 1, HMILDs consists of three components as follows: 1) the Candidate Selector component (CS), 2) the Instance Matching component (IM) and 3) the Candidate Expander component (CE). CS retrieves candidate instances from an anchor data set by using a set of keywords to generates a set of candidate pairs. IM verifies whether generated candidate pairs are correctly match or not by using a machine learning technique based on similarity vectors between instances. CE heuristically expands the search space from an anchor data set to another LOD data set in order to find other candidate instances for non-matched instances and re-generates a new candidate pair. The details of the components are as follows.

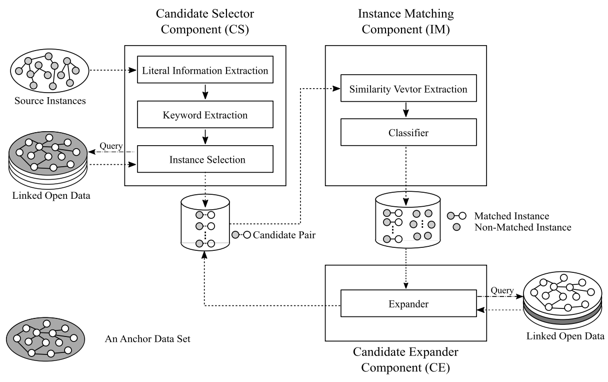


Fig. 1 The diagram of HMILDs

4.1 Candidate Selector

As shown in Fig. 1, source instances are passed into CS. In CS, there are three modules: 1) the literal information extraction module, 2) the keyword extraction module and 3) the instance selection module. The literal information extraction module extracts description of a source instance. The keyword extraction module creates a set of keywords from description of the source instance. The instance selection module finds candidate instances in the anchor data set by using the set of keyword and then pairs the candidate instances with the source instance as candidate pairs. The result of CS is a set of candidate pairs.

4.1.1 Literal Information Extraction

The literal information module extracts description of the instance referred as literal information for each source instance. There are many studies [12], [14], extracting the literal information by considering rdfs:label and some other common properties [15]. Although the common properties are widely used to describe many LOD instances, due to heterogeneous problem, some instances might not contain those properties. Limiting extracting literal information to some properties might miss some useful information. Therefore, in the literal information extraction module, all properties are considered as literal information.

The literal information is divided into two types. Two types of literal information are a short literal string l_s and a long literal string l_l. l_s is a string, of which the length equals one phrase, whereas l_l is a string, of which the length is greater than one phrase. Considering the characteristic of l_s, l_s uses to represent specific information of an instance since a short string usually behave as a label of the instance or a concise description of the instance. In contrast with l_s, l_l generally describes greater detail of an instance. Therefore, l_s carries much more essential information than l_l. Nevertheless, in case that l_s of instances causes an ambiguity, l_l could help to disambiguate between instances. For example, given the instance, db:Barack_ Obama[†],

db:Barack_Obama	foaf:surname	"Obama"@en
	rdfs:comment	"Barack Hussein Obama II (born August 4, 1961)...."@en

"Obama" is treated as l_s because its length equals one phrase. "Barack Hussein Obama II (born August 4, 1961) ...." should be considered as l_l because its length is greater than one phrase. In the example, The l_s "Obama" could not disambiguate between db:Barack_Obama and db:Michelle_Obama[††] because both instances are referred to "Obama". Nonetheless, the l_l "Barack Hussein Obama II (born August 4, 1961) ...." could help to distinguish between the instances. Therefore, HMILDs does not only consider l_s but also consider l_l. Consequently, the results of this module are l_s and l_l of an instance.

4.1.2 Keyword Extraction

In the keyword extraction module, l_s and l_l are used to create a set of keywords. Since the characteristic between l_s and l_l is different, the different methods for extracting keywords are applied.

For l_s mostly contains critical information of an instance, omitting some words might not be able to represent an identity of instance. Each word in l_s therefore is selected as a keyword. Furthermore, the N-gram technique is also applied to capture co-occurrence words. For example, given l_s as "San Francisco", if we consider the words, "San" and "Francisco", separately, it might not be able to represent the exact meaning of "San Francisco". Consequently, the words generated by the N-gram technique is employed as keywords to cope with such characteristic.

For l_l, the same strategy as l_s cannot be applied because l_l comprises a lot of words. It therefore would be better to select some words. Name Entity Recognition (NER) technique [16] is employed in order to select keywords from l_l. Usually name entities such as person, location, organization name are highly relate to an instance. With the NER technique, a small set of keywords could be created. After acquiring keywords from l_s and l_l, a set of keywords is constructed by combing all keywords together.

4.1.3 Instance Selection

The instance selection module generates candidate pairs between a source instance and an anchor instance by using a set of keywords. Each keyword is used to retrieve anchor instances, which contain the same keyword. After that, the anchor instance are paired with the source instance as the candidate pair.

4.2 Instance Matching

In IM, each candidate pair from CS is verified whether it is a correct match or not by using a similarity vector. Consequently, the results of IM are match pairs and non-match instances. In IM, there are two modules: 1) the similarity vector extraction module and 2) the classifier module. In the

[†]http://dbpedia.org/page/Barack_Obama

[††]http://dbpedia.org/page/Michelle_Obama

similarity vector extraction module, a similarity vector of a candidate pair is computed by various similarity metrics. The classifier module then classifies the candidate pair by using its similarity vector. When the classifier module classifies that the candidate pair is a correct match, the source instance of the candidate pair immediately map to its paired instance. Such relation could be utilized to access expanded instances later. Otherwise, the candidate pair is unpaired as a non-match instance.

4.2.1 Similarity Vector Extraction

In the similarity vector extraction module, a candidate pair is passed to various similarity metrics to compute a similarity vector between the instances. In the LOD cloud, usually instances are heterogeneous and some of them might be distorted and ambiguous [17]. As a result, limiting similarity metrics to a few metrics might not overcome such problems. In the similarity vector extraction module, many kinds of similarity metrics therefore are used in HMILDs.

For the similarity metrics, the six similarity metrics: Term Frequency-Inverse Document Frequency cosine similarity (TF-IDF) [18], Jaro-Winkle similarity metric [19], Edit Distance similarity metric [20], Count similarity metric [12], IDF similarity metric [12] and TopIDF similarity metric [12], are used. These similarity metrics are conventional similarity metrics for representing the similarity between documents. In HMILDs, an instance could be viewed as a document because we treat all literal information of instances as string. To create a document from an instance, all strings of an instance are appended together as a document. Consequently, we could apply these similarity metrics as the same manner of the conventional document similarity.

In addition, we also introduce two novel similarity metrics: the CommonKeyword similarity metric and the CandidateHits similarity metric.

The CommonKeyword similarity metric aims to capture the similarity between a set of keywords of instances. Based upon the characteristic of a set of keywords, instances sharing a lot of keywords have a high chance to be a match pair, because the instances of such candidate pairs intend to describe the same thing. Consequently, we assume that the more keywords instances of a candidate pair share, the more likely they are matched. To capture this characteristic, the CommonKeyword similarity metric is calculated by passing sets of keywords between instances of a candidate pair to the Jaccard Similarity [21].

The CandidateHits similarity metric is to represent how many times the candidate pairs are generated. During generating candidate pairs, some candidate pairs might be generated more than once. For example, different keywords might retrieve the same candidate instance for generating the candidate pair. We therefore assume that the candidate pair, which is generated frequently, is likely to match, because the relation between such instance and the source instance is highly correlate. Based on this concept, the CandidateHits similarity metric is derived as shown in Eq. (1),

$$CandidateHits(c) = \frac{n(c)}{\sum_{i \in C} n(i)} \tag{1}$$

where $n(c)$ and $n(i)$ is how many times the candidate pair c and the candidate pair i are generated and C is a set of candidate pairs.

Apart from eight similarity metrics, we also consider types of literal information as a factor of combination of similarity metrics. Ignoring type of literal information might miss some description of an instance. We categorize literal information into three types: 1) l_s, 2) l_l and 3) combining l_s and l_l together. In HMILDs, three documents regard types of literal information are extracted for each instance. Then, similarity metrics are applied for each document of the instance. However, a dimensional length of our feature vector is 22-dimensional feature vector. Since the CandidateHits similarity metric does not relate to a type of literal information of instances, a candidate pair applies this similarity metric only once.

4.2.2 Classifier

In the classifier module, a machine learning method is utilized to create a classifier C for determining whether the candidate pair is correct match or not. To create the classifier C, similarity vectors of candidate pairs and their label indicating the class of candidate pair are used. If the classifier C classifies the candidate pair as a correct match, the source instance of the candidate pair will be mapped to the LOD instance of the candidate pair. Otherwise, the candidate pair is unpaired as a non-match instance.

4.3 Candidate Expander

For non-match instances, CE expands the search space for finding other candidate instances in other LOD data sets to generate new candidate pairs. In CE, only the expander module is installed. The expander module enables HMILDs to traverse through the LOD cloud by using link properties, in particular the owl:sameAs property, in order to discover a new candidate instance for re-generating a new candidate pair.

Generally, adjacent instances of an instance carries relevant information about the instance. We therefore assume that adjacent instances could be utilized to discover a new candidate instance. The basic idea is to start with a non-matched instance and gradually expand the search space to its adjacent instances. Then, such adjacent instances continue to expand the search space to their adjacent instances. The expansion procedure is done repeatedly until a candidate instance of the non-matched instance could be discovered or there are no any adjacent instances to expand the search space.

Although we could gradually expand an source instance to an expanded instance by using adjacent instances and link properties, the expanded search space could grow up dramatically when expanding into higher depth level [13]. The depth level is measured by how many hops from the instance to the other instance via adjacent instances. According to this problem, a simple heuristic function is introduced by assuming that the search space should be expanded to an expanded instance if the expanded instances share at least a keyword with instances in traversed path of adjacent instances in the source data set [13]. Based on the preliminary experiment [13] with the maximum depth level set at 3, the expander module generates averagely 214,885 candidate pairs per an instance, while using the simple heuristic function the expander module establishes 203 candidate pairs per an instance. The heuristic function in the work [13] could reduce a number of candidate pairs. Still, the work [13] generates enormous useless candidate pairs, which have no chance to be a correct match pair.

In this article, a new heuristic function for generating candidate pairs is introduced. The aim of the heuristic function is to limit the expanded search space so that a less number of candidate pairs and a less number of useless candidate pairs would be generated.

In HMILDs, the keyword score is proposed as the new heuristic function. In the keyword score, two basic assumptions are made. First, the more instances share keywords, the more they are likely to match. This assumption is similar to the CommonKeyword similarity metric. Second, keywords from adjacent instances in the different depth level are distinct. Keywords of adjacent instances in the lower depth level could more highly related to the instance than keywords of adjacent instances in the higher depth level. Consequently, it could be assumed that the lower the depth level of keywords to the instance is, the more important the keyword is. Based upon two assumption above, a keyword score of an instance could be derived as shown in Eq. (2),

$$KeywordScore_s(e) = \frac{\sum_{i=0}^{d} w^i \cdot \frac{|keyword(e) \cap \cup_{x \in Adj(s)_i} keyword(x)|}{\sum_{x \in Adj(s)_i} |keyword(x)|}}{\sum_{i=0}^{d} w^i} \tag{2}$$

where e is an expanded instance, $keyword(x)$ is a function returned keywords of the instance x, d is the maximum depth level for expanding the search space, w is a parameter for weighting necessity of keywords at the different depth level and $Adj(s)_i$ is a set of adjacent instances of s at the depth level i. For example, given the scenario in Fig. 2, the instance "Baron Papanoida" in the source data set is s and the instance "Star war iii Revenge of the Sith" in the expanded data set is e. The instances, "Star war iii Revenge of the Sith" and "George Lucas", in the source data are instances in $Adj(s)_1$.

The aim of keyword score is to capture the correlation between the source instance s and the expanded instance e. If the keyword score is high, the correlation between instances will be high. We assume that the expander module should expand the search space to the high correlated instance because there is high probability to discover an identical instance.

In Algorithm 1, the algorithm of the expander module is expressed. The idea of the algorithm is to find a pair

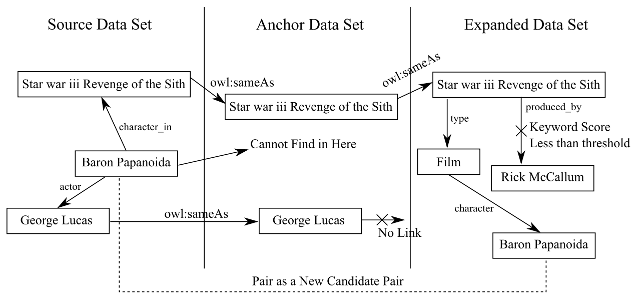


Fig. 2 An example of the expanded search space from One Data Set to Other Data Sets.

Algorithm 1 Expander Module for Generating New Candidate Pairs

```
Input: $NI_s$ (Set of non-match instances)
Output: $C$ (Set of Candidate pair)
 1: $C \leftarrow \emptyset$
 2: $Q \leftarrow \emptyset$      # $Q$ is a queue of instances, which will explore and expand
 3: foreach $s \in NI_s$ do
 4:   $Q \leftarrow s$
 5:   while $Q \neq \emptyset$ do
 6:     $p \leftarrow Q.Dequeue()$                 # Get the first instance of $Q$
 7:     $N \leftarrow Adj(p)$          # Get the adjacent instances of instance $p$
 8:     foreach $e \in N$ do
 9:       if $KeywordScore_s(e) > \delta$ then
10:         $Q \leftarrow e$              # Add to $Q$ for exploring next round
11:         $C \leftarrow (s, e) \cup C$            # Pair candidate between $s$ and $e$
12:       end if
13:     end foreach
14:     $Q \leftarrow sameAsLink(p)$           # Get other instances linked to $p$
                                     # by owl:sameAs and store in $Q$
15:   end while
16: end foreach
17: return $C$
```

of a source instance and an adjacent instance, of which the keyword score is greater than the threshold δ. Then, the algorithm gradually expands the search space by using the owl:sameAs property of that adjacent instance in order to reach an expanded instance and pair the expanded instance with the source instance as a new candidate pair.

In Fig. 2, an example of the algorithm is illustrated. Given the source instance "Baron Papanoida", this instance does not match any instances in the anchor data set; in consequence, it is a non-matched instance. However, the instance "Baron Papanoida" has some adjacent instances: "Star war iii Revenge of the Sith" linking to "Baron Papanoida" by the character_in relation and "George Lucas" linking to "Baron Papanoida" by the actor relation, and they successfully map to the anchor data set. Utilizing the owl:sameAs property, the expander module could traverse through the expanded data set via the owl:sameAs property among the source instance, the anchor instance and the expanded instance. Then, the expander module expands and locally searches the expanded instance "Star war iii Revenge of the Sith" in order to discover a new candidate instance for the source instance "Baron Papanoida". Then, the new candidate pair between the source instance "Baron Papanoida" and the expanded instance "Baron Papanoida" could be acquired. In case that the keyword score of expanded instances is less than the threshold, the instance is removed from the expanded search space in order to limit the range of the search space. In Fig. 2, the instance "Rick McCallum" is removed, since its keyword score is less than the threshold δ.

5. Experiments

5.1 Experimental Setup

To evaluate the framework, four experiments are conducted. The first experiment investigates contribution of our novel similarity metrics for IM. The second experiment evaluates the performance of CS and CE for mapping instances to multiple LOD data sets. The third experiment investigates parameters for the heuristic function of HMILDs. The forth experiment evaluates the performance of HMILDs.

Instances from Ontology Alignment Evaluation Initiative 2012 (IM@OAEI 2012) track [22] are used as source instances. Then, the source instances are aligned to LOD instance by a domain expert in order to construct ground truth for experiments. All experiments are conducted by using this dataset. Due to a large number of LOD data sets, manual aligning source instances to all data sets is impossible. Therefore, instances in two prominent LOD data sets, DBpedia and Freebase, are selected to align to source instances for constructing ground truth. DBpedia [23] is also chosen as the anchor data set because DBpedia is well-known as the hub of the LOD cloud [24]. Based upon the preliminary experiment, we could find that only **90.36%** of our source instances could be manually mapped to DBpedia instances. This result shows the solid evidence conforming to our research statement, where only one data set might not be enough for mapping instances. Freebase [25] is selected as the expanded data set because more than million links of DBpedia are connected to Freebase. Furthermore All of our source instances, which do not contain in DBpedia, could be found in Freebase.

In HMILDs, there are many parameters. To conduct the experiments, the parameters are set as follows. In the keyword extraction module, N for the N-gram technique are set as 1, 2, 3 and n respectively, where n is the length of a considered string. For NER system, Stanford Named Entity Recognizer [26], is used. In the instance selection module, the DBpedia Lookup Service interface [27] is used to gather candidate instances from DBpedia. In the instance matching module, the Support Vector Machine (SVM), LIBSVM [28], is used in this component as the classifier to classify a candidate pair as the same manner of the work [29]. To build the classifier, candidate pairs generated by HMILDs are manually labeled to their corresponded class. Then, such label together with their similarity feature vectors are used to create parameters of the classifier.

Table 1 Summary of similarity metrics for each method

Method	Similarity Metrics							
	TF IDF	Jaro Winkler	Edit Distance	Count [12]	IDF [12]	TopIDF [12]	Common Keyword	Candidate Hits
TF-IDF	✓							
TF-Jaro [19]	✓	✓						
RiMOM [20]	✓		✓					
Rong et al. [12]	✓		✓	✓	✓	✓		
Combined [13]	✓	✓	✓	✓	✓	✓		
HMILDs-CH	✓	✓	✓	✓	✓	✓	✓	
HMILDs-CK	✓	✓	✓	✓	✓	✓		✓
HMILDs	✓	✓	✓	✓	✓	✓	✓	✓

Table 2 The results of each method

Method	Precision	Recall	F-Measure
TF-IDF	0.89204	0.76874	0.82276
TF-Jaro [19]	0.91825	0.79228	0.84760
RiMOM [20]	0.92389	0.82998	0.87256
Rong et al. [12]	**0.95569**	0.81925	0.88100
Combined [13]	0.94999	0.85782	0.90013
HMILDs-CH	0.95243	0.87349	0.90970
HMILDs-CK	0.94799	0.88237	0.91202
HMILDs	0.95202	**0.91008**	**0.92949**

5.2 Experiment 1

Experiment 1 is designed to investigate contribution of our two novel similarity metrics, CommonKeyword and CandidateHits, for IM. Furthermore, various similarity metrics are compared to find the suitable combination of the similarity metrics for the instance matching component. In the instance matching problem, many combination of a similarity metrics have been proposed [12], [13], [19], [20]. In the experiment, we compare combination of similarity metrics with combination of our similarity metrics. Combination of similarity metrics of our approach, including HMILDs, HMILDs without the CandidateHits similarity metric (HMILDs-CH) and HMILDs without the CommonKeyword similarity metric (HMILDs-CK), and other studies [12], [13], [19], [20] are summarized as shown in Table 1.

In the experiment, 10-fold cross-validation technique is applied to evaluate each combination of similarity metrics. Precision, Recall and F-Measure are employed to measure the performance of the results.

The experimental results are listed in Table 2. Considering the results in Table 2, we could notice that Rong's method provides the highest precision at 0.956, while HMILDs gives the best recall and the best F-measure at 0.910 and 0.929 respectively. Although HMILDs provides less precision than Rong's method, the highest result of the F-measure still acquired from HMILDs.

In order to deeply investigate the results, HMILDs is compared against other methods by the t-testing method. The significance level is set at 0.05. It turns out that although combination of our similarity metrics gives the lower precision result than Rong's method [12], the difference of the results is not significant. Nevertheless, the paired t-test's results of the recall and the F-Measure indicate that HMILDs is significantly different from other methods excepting HMILDs-CH and HMILDs-CK, which include the

CommonKeyword similarity metric or the CandidateHits similarity metric. To further investigate contribution of each similarity metric, the CommonKeyword similarity and the CandidateHits similarity, the t-test's results of the recall and the F-Measure between HMILDs-CH, HMILDs-CK and other methods, excepting HMILDs, are conducted. The t-test results turn out that for HMILDs-CH there is no significance when comparing with Combined [13], while for HMILDs-CK, there is no significance when comparing with Combined [13] and RiMoM [20]. Based upon this investigation, the CommonKeyword similarity metric together with the CandidateHits similarity metric could provide significant contribution for improving the recall result and the F-Measure result of IM while they do not affect the precision result.

5.3 Experiment 2

Experiment 2 is to investigate the contribution of CE for mapping instances to many LOD data sets and the contribution of keywords in CS for discovering identical instances. In the experiment, to fairly evaluate CS and CE, all generated candidate pairs are manually verified whether they are correct match or not so that the effect of IM could be avoided.

There are four setting in the experiment. The first setting is the Lookup [30] with DBpedia as a baseline approach. The Lookup approach retrieves candidate instances by using only a label of an instance as a keyword. The second and the third settings are CS with two different data sets, DBpedia and Freebase respectively. The fourth setting is CS with CE (CS + CE). For CS + CE, the anchor data set is DBpedia and the expanded data set is Freebase. Three metrics: selecting, non-selecting and missing, are used to evaluate the components in the experiment. The selecting metric measures the percentage of existing identical instances between the source data set and the LOD data set retrieved by the component for generating candidate pairs. The non-selecting metric measures the percentage of existing identical instances between the source data set and the LOD data set, which could not be retrieved by the component for generating candidate pairs. The missing metric represents the percentage of non-existing identical instances between the source data set and the LOD data set.

Considering the results in Table 3, CS using the data set DBpedia outperforms Lookup. Even though both CS and Lookup utilize a set of keywords for retrieving an instance in DBpedia, the main difference is a method to obtain a set of keywords. Lookup selects only a label of an instance as a keyword, while our CS generates a set of keywords by using various strategies. Therefore, The result confirms that our keywords in CS are helpful to discover identical instances.

Table 3 The results of generating candidate pairs

Setting	Data Set	Selecting	Non-Selecting	Missing
Lookup [30]	DBpedia	84.30%	6.06%	9.64%
CS	DBpedia	88.71%	1.65%	9.64%
CS	Freebase	95.04%	4.96%	0.00%
CS + CE	DBpedia Freebase	97.80%	2.20%	0.00%

The results of the experiment are listed in Table 3. In the Table 3, two main contribution for CE are founded. Firstly, comparing the missing result of CS using DBpedia with CS + CE, the missing result reduces from 9.64% to 0.0%. This reduction significantly shows that CE could discover some missing instances in one data set by using expanded data set. This result conforms to the assumption of this research, where mapping instances to one LOD data set is not enough. Secondly, considering the results between CS and CS + CE, the selecting result of CS + CE is greater than the selecting result of CS with any data sets. This results shows that CE could help to discover more identical instances because it can successfully map instances to many data sets. Based upon two contribution of CE, we conclude that CE greatly contributes to map instances to the LOD cloud.

5.4 Experiment 3

In Experiment 3, parameters in the heuristic function are studied to investigate the trade-off between the ability to limit the expanded search space and the ability to map instances to the LOD cloud.

Two parameters are studied in the experiment. The first parameter is the threshold δ for limiting the expanded search space. The threshold δ in the experiment is varied from 0.0 to 0.5 and increases each step by 0.05. The second parameter is the weighting w for computing the keyword score. In the experiment, the weight w at 0.0, 0.25, 0.5 and 1.0 are investigated respectively.

To fairly evaluate the performance and the goodness of the heuristic function in the framework, the work [13] is selected as the baseline for the experiment. The framework in the study [13] is mostly similar to HMILDs in this article; however, the heuristic function of the expander module is different. In the experiment, the 10-fold cross validation technique is performed to evaluate the results. Based upon our preliminary experiment [13], we statically set the depth level d at 3, since it could be sufficient enough to reach candidate instances and does not allow the expander module to explore the large search space.

Three evaluation metrics: 1) the average number of generated candidates, 2) the average number of useless candidates and 3) the coverage percentage of mapping instances to the LOD cloud, are used to evaluate the ability to map instances to the LOD cloud and the ability to limit the expanded search space in aspects of quantity and quality.

The average number of generated candidates is used to evaluate the ability to limit the expanded search space in the aspect of quantity. The expanded search space directly relates to a number of generated candidate pairs. If the expanded search space is very large, a number of generated candidate pairs will also become greater. Consequently, the ability to limit the expanded search space in the aspect of

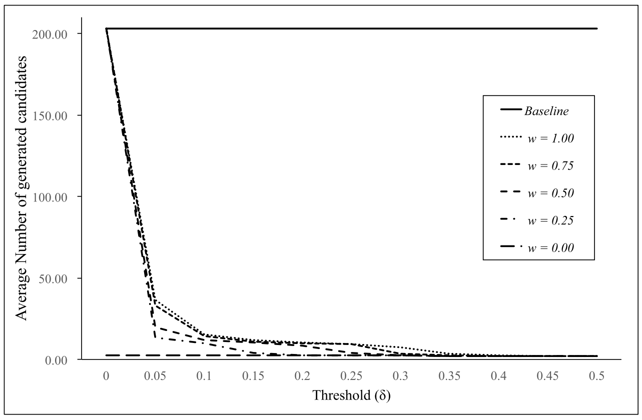


Fig. 3 Average number of generated candidate pairs of the baseline and the framework with different weight w varied by the threshold δ

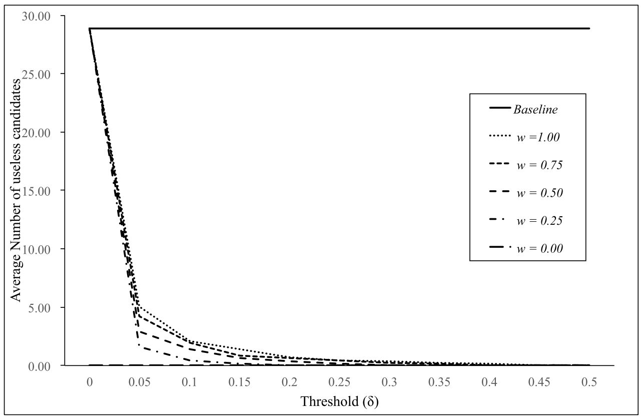


Fig. 4 Average number of useless generated candidate pairs of the baseline and the framework with different weight w varied by the threshold δ

quantity could be observed through an average number of generated candidate pairs.

The average number of useless candidates is employed to measure the quality of the ability to limit the expanded search space. Although many useless candidate pairs are generated, such candidate pairs could not improve any coverage for mapping instances to the LOD cloud. Therefore, the quality of the ability to limit the expanded search space could be observed via an average number of useless candidate pairs.

The coverage percentage of mapping instances to the LOD cloud is used to evaluate the ability to map instances to the LOD cloud. If the percentage of mapping instances to the LOD cloud is high, it could be inferred that the ability to map instances to the LOD cloud is also high.

The results of the experiment are illustrated in Figs. 3–5. In each figure, the x-axis is the threshold δ, while the y-axis is the result measured by each metric. Each line in the figure represents the result with the different weight configuration.

In the Figs. 3–4, the results show that the threshold δ directly influences the ability to limit the expanded search

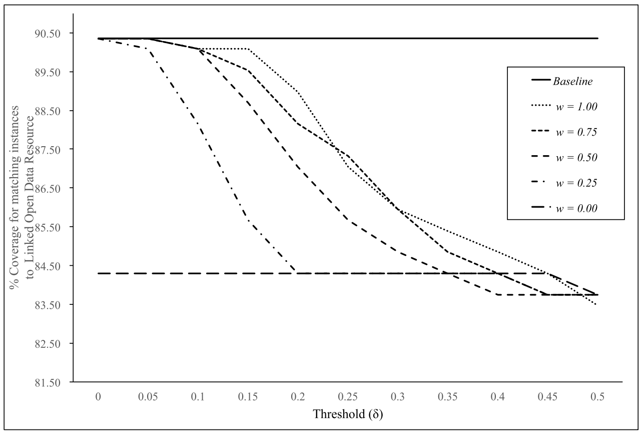


Fig. 5 Coverage for mapping instances to the LOD cloud of the baseline and the framework with different weight w varied by the threshold δ

space. At the threshold $\delta = 0.05$, the dramatic reduction of a number of generated candidate pairs and reduction of a number of useless candidate pairs could be observed. Therefore, the threshold δ directly plays an important role in the ability to limit the expanded search space.

Considering effect of the weight w in Figs. 3–4, we could observe that the weight w contribute to the ability to limit the expanded search space at the threshold δ set at 0.05. When the weight w is reduced, the heuristic function tends to generate less candidate pairs and useless candidate pairs. This characteristic happens because when the weight w reduces, the priority of the keyword is given to the keyword that closes to the source instance. As a result, the heuristic function will eliminate useless candidate pairs.

In the Fig. 5, when the threshold δ increases, the ability to map instances to the LOD cloud is decreased as we could observe from the reduction of the coverage number of mapping source instances to the LOD cloud. For the weight w, considering the results in different weight w in Fig. 5, the different weight w could differently provide the coverage number of mapping source instances to the LOD cloud. However, the effect of the weight w could be governed by the threshold δ when the threshold δ is too large.

According to the experimental result, the threshold δ and the weight w contribute to the ability to limit the expanded search space and the ability to map instances to the LOD cloud. The best configuration of the weight w and the threshold δ for balancing between the ability to limit the expanded search space and the ability to map instances to the LOD cloud are achieved, when the weight w is set at 0.5 and the threshold δ is set at 0.05. With such configuration, HMILDs could produce only 9.73% of candidate pairs of the baseline and 10.05% of useless candidate pairs of the baseline, whereas the highest coverage result of mapping source instances to the LOD cloud at 90.36% is still reached as same as the baseline.

Table 4 The results of the framework comparing with the baseline

Approach	Precision	Recall	F-Measure	Avg. number of Generated Candidates	Avg. number of useless generated candidates	% Coverage
Baseline [13]	0.913	0.919	0.914	203.03	28.87	90.36
HMILDs	0.917	0.919	0.917	11.74	2.90	90.36

5.5 Experiment 4

In Experiment 4, the heuristic function of HMILDs is investigated in three aspects. The first aspect is overall performance. The second aspect is the effectiveness to limit the expanded search space. The third aspect is as the efficiency for mapping instances to the LOD cloud.

In the experiment, *Baseline* from the study [13] is used as the benchmark. The framework [13] is mostly similar to HMILDs; however, the heuristic function of the expander module is different. In the experiment, the parameters of HMILDs are set as follows. The weight w is set at 0.5. The threshold δ is set at 0.05. The depth d is set at 3. To evaluate the results the 10-fold cross validation technique is performed. All results are reported in Table 4.

In the first aspect, three standard metric: precision, recall and F-measure are employed to evaluate the performance of HMILDs, when the heuristic function is installed. As shown in Table 4, HMILDs provides the similar precision result, the similar F-Measure result and the same recall result, when comparing with *Baseline*. Consequently, in the first aspect regarding the performance, the heuristic function could not affect any performance. This result indicates that even though the expanded search space is reduced due to the heuristic function, HMILDs could still obtain the similar performance as Baseline.

In the second aspect, the average number of generated candidates and the average number of useless generated candidates are used to analyze this aspect. The expanded search space directly relates to a number of generated candidate. If the expanded search space is very large, a number of generated candidate pairs will be plentiful. Consequently, we could observe the effectiveness of the heuristic function for limiting the expanded search space via the average number of generated candidates and the average number of useless generated candidates. As shown in Table 4, the results indicate that our heuristic function helps to limit the expanded search space due to the reduction of an average number of generated candidate pairs. Furthermore, we could also observe that *Baseline* generates more useless candidate pairs than HMILDs. Due to the reduction of generated candidate pairs and useless candidate pairs, it could be concluded that the heuristic function of HMILDs could limit the expanded search space effectively.

In the third aspect, the percentage of coverage of mapping instances to the LOD cloud (%Coverage) is measured to represent the efficiency of the heuristic function for mapping instances to the LOD. Although HMILDs could generate less generated candidate pairs and useless candidate pairs than *Baseline* in the second aspect, we also need to consider the efficiency of the heuristic function in the third aspect because the main purpose of the research is to map instances to the LOD cloud as many as possible. As shown in Table 4, HMILDs provide the same percentage of coverage of mapping instances to the LOD cloud to *Baseline*. Therefore, the heuristic function of HMILDs does not affect the percentage of coverage of mapping instances to the LOD cloud even though the expanded search space is reduced.

Based upon the results in all aspects, HMILDs outperforms *Baseline* in the second aspects and does not provide worse results than *Baseline* in any aspects. Therefore, This experiment indicates that the heuristic function of HMILDs could effectively limit the expanded search space, while still maintains the efficiency to map instances to the LOD without affecting any performances.

6. Conclusion

In this article, HMILDs is introduced. The experimental results showed that HMILDs could successfully map instances to LOD data sets. Furthermore, the heuristic function of HMILDs could generate fewer candidate pairs and significantly reduce the number of useless candidate pairs better than the other work without degrading performance.

Although HMILDs could discover candidate instances across LOD data sets, some of such candidate instances could not be successfully paired with source instances. Due to the heterogeneous problem in LOD data sets, IM in HMILDs might not be able to correctly verify all candidate pairs generated from the expanded search space whether they are a correct match or not. In the future work, we therefore plan to integrate other powerful instance matching techniques into IM of HMILDs to deal with the heterogeneous problem of LOD data sets.

References

[1] T. Berners-Lee, “Linked Data - Design issues,” http://www.w3.org/-DesignIssues/LinkedData.html, 2006.

[2] G. Klyne and J. Carroll, “Resource Description Framework (RDF): Concepts and Abstract Syntax,” W3C Recommendation, http://www.w3.org/TR/rdf-concepts/, 2004.

[3] C. Bizer, T. Heath, and T. Berners-Lee, “Linked data - the story so far,” International Journal on Semantic Web and Information Systems, vol.5, no.3, pp.1–22, 2009.

[4] R. Cyganiak and A. Jentzsch, “The Linking Open Data cloud diagram,” Available: http://www.lod-cloud.net/, 2014.

[5] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, “Discovering and maintaining links on the web of data,” In Proceedings of the 8th International Semantic Web Conference, LNCS, vol.5823, pp.650–665, Springer, Heidelberg, 2009.

[6] 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 Proceedings of the 6th Workshop on Ontology Matching, pp.85–113, 2011.

[7] X. Niu, S. Rong, Y. Zhang, and H. Wang, "Zhishi.links results for OAEI 2011," In Proceedings of the 6th Workshop on Ontology Matching, pp.220–227, 2011.

[8] W. Hu, J. Chen, and Y. Qu, "A self-training approach for resolving object coreference on the semantic web," In Proceedings of the 20th International Conference on World Wide Web, pp.87–96, ACM, 2011.

[9] S. Araujo, D.T. Tran, A.P. de Vries, and D. Schwabe, "SERIMI: Class-based disambiguation for effective instance matching over heterogeneous web data," base, vol.27, no.5, pp.1397–1440, 2015.

[10] K. Nguyen, R. Ichise, and B. Le, "SLINT: A schema-independent linked data interlinking system," In Proceedings of the 7th Workshop on Ontology Matching, pp.1–12, 2012.

[11] K. Nguyen, R. Ichise, and B. Le, "Interlinking Linked Data Sources Using a Domain-Independent System," In Proceedings of the 2nd Joint International Semantic Technology Conference, vol.7774, pp.113–128, 2013.

[12] S. Rong, X. Niu, E.W. Xiang, H. Wang, Q. Yang, and Y. Yu, "A Machine Learning Approach for Instance Matching Based on Similarity Metrics," In Proceedings of the 11th International Semantic Web Conference, vol.7649, pp.460–475, 2012.

[13] N. Kertkeidkachorn, R. Ichise, A. Suchato, and P. Punyabukkana, "An Automatic Instance Expansion Framework for Mapping Instances to Linked Data Resources," In Proceedings of the 3rd Joint International Semantic Technology Conference, pp.380–395, 2014.

[14] M. Vickers, Ontology-based free-form query processing for the semantic web, Master's thesis, Brigham Young University, Provo, Utah, 2006.

[15] B. Ell, D. Vrandečić, and E. Simperl, "Labels in the Web of Data," In Proceedings of the 10th International Semantic Web Conference, vol.7031, pp.162–176, 2011.

[16] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification. In Linguisticae Investigationes," vol.30, no.1, pp.3–26, 2007.

[17] E. Ruckhaus, M.-E. Vidal, S. Castillo, O. Burguillos, and O. Baldizan, "Analyzing Linked Data Quality with LiQuate," In Proceedings of European Semantic Web Conference, vol.8798, pp.488–493, 2014.

[18] W.W. Cohen, "Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity," In Proceedings of the International Conference on Management of Data, vol.27, no.2, pp.201–212, 1998.

[19] W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string metrics for matching names and records," In Proceedings of the Workshop on Data Cleaning and Object Consolidation, pp.73–78, 2003.

[20] J. Li, J. Tang, Y. Li, and Q. Luo, "RiMOM: A dynamic multistrategy ontology alignment framework," IEEE Transactions on Knowledge and Data Engineering, vol.21, no.8, pp.1218–1232, 2009.

[21] T. Pang-Ning, S. Michael, and K. Vipin, Introduction to Data Mining, First Ed., Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2005.

[22] "Ontology Alignment Evaluation Initiative 2012 Campaign," http://oaei.ontologymatching.org/2012/, 2012.

[23] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A Nucleus for a Web of Open Data," In Proceedings of 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference, vol.4825, pp.722–735, 2007.

[24] G. Kobilarov, C. Bizer, S. Auer, and J. Lehmann, "DBpedia-A Linked Data Hub and Data Source for Web and Enterprise Applications," In Proceedings of the 18th World Wide Web Conference, pp.1–3, 2009.

[25] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," In Proceedings of the 28 ACM SIGMOD International Conference on Management of data, pp.1247–1250, 2008.

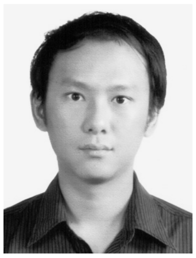
[26] J.R. Finkel, T. Grenager, and C. Manning, "Incorporating Non-Local Information into Information Extraction Systems by Gibbs Sampling," In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, pp.363–370, 2005.

[27] M. Watson, "Practical Semantic Web and Linked Data Applications," http://www.markwatson.com/opencontent-data/book-java.pdf, 2011.

[28] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," ACM Transactions on Intelligent Systems and Technology, vol.2, no.3, pp.1–27, 2011.

[29] R. Ichise, "Machine Learning Approach for Ontology Mapping using Multiple Concept Similarity Measures," In Proceedings of the 7th IEEE/ACIS International Conference on Computer and Information Science, pp.340–346, 2008.

[30] F. Caimi, Ontology and Instance Matching for the Linked Open Data Cloud, Master Thesis of University of Illinois at Chicago, 2012.



Natthawut Kertkeidkachorn received his B.Eng and M.Eng degree in computer engineering from Chulalongkorn University, Bangkok, Thailand in 2011 and 2013 respectively. He is currently a Ph.D candidate at Sokendai (The Graduate University for Advanced Studies) in Japan. His research interests include semantic web, machine learning and natural language processing.

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