# **RiMOM-IM Results for OAEI 2014**

Chao Shao, Linmei Hu, Juanzi Li

Tsinghua University, Beijing, China. {shaochao,hulinmei,ljz} @ keg.tsinghua.edu.cn

**Abstract.** This paper presents the results of RiMOM-IM in the Ontology Alignment Evaluation Initiative (OAEI) 2014. We only participated in IM@OAEI2014. We first describe the overall framework of our matching System (RiMOM-IM); then we detail the techniques used in the framework for instance matching. Last, we give a thorough analysis on our results and discuss some future work on RiMOM-IM.

## **1** Presentation of the system

Recently, a number of ontological knowledge bases have been built and published, such as DBpedia[1]. , YAGO [2], Xlore [3], etc. Some published knowledge bases are domain specific ones that cover facts within one domain, such as movie, music and geography; some other ones are cross-domain knowledge bases that contain various kinds of information in different domains. Usually, knowledge about one object may be contained in different knowledge bases. For example, both YAGO and elvisPedia contain 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 know more about "Elvis Presley", we have to search his information in different knowledge bases. Therefore, there is a growing need to align different knowledge bases so that we can easily get more complete knowledge about things that we are interested in.

A lot of work has already been done for aligning ontological knowledge bases. Previous researches focus on aligning the schema elements (i.e. concepts and properties) in knowledge bases, which is called ontology matching. Most recently, the problem of matching instances in different knowledge bases has attracted increasing interest. Many instance matching approaches have been proposed. Our system is proposed for largescale instance matching. There are two major techniques in the existing approaches to speed up the instance matching process: blocking and iterative matching. Blocking is to index the instances in two knowledge bases separately and then select the instances having the same keys as candidate instance pairs. Iterative matching is to find 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 rest instances in the following iterations. Although the above two techniques are very helpful to large-scale instance matching, there are still several challenging problems which are not well addressed. First, since usually only literal values in RDF triples are used as indexing keys for blocking, the set of candidate instance pairs to be compared 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 [4]. 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 high accuracy of instance matching.

#### 1.1 State, purpose, general statement

This section describes the overall framework of RiMOM-IM. The overview of the instance matching system is shown in Fig. 1. The system includes five modules, i.e., *Initial Interactive Configuration, Candidate Pair Generation, Matching Score Calculation, Instance Alignment* and *Validation*. The annotated numbers in the figure show the sequences of the process. We illustrate the process as follows.



Fig. 1. Framework of RiMOM-IM

- 1. The system begins with *Initial Interactive Configuration*, which allows users to configure system with needed modules in the following process and their parameters.
- 2. We conduct data preprocessing, such as unifying data formats for the values of some predicates.
- 3. We proceed blocking which consists in using inverted indexing to generate *candidate set* and *unique instance sets*.
- 4. For each pair in the *candidate set*, we compute 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. We generate a priority queue by sorting the final scores in a descending order.

- 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 find new candidate pairs and new unique instances, thus updating candidate set and unique instance sets. Correspondingly, 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 newly aligned instance pair, we will generate new unique instances, which will be taken as input to step 5. If there is no new aligned pair, we continue step 7.
- 7. If Validation module is chosen in step 1, we will conduct validation on all aligned pairs. Otherwise, terminate.

### 1.2 Specific techniques used

This year we only participate in the IM@2014 track. We will describe specific techniques used in this track.

**Data Preprocessing**: First, we translate all the languages used in the whole datasets to English by using google translator. Then we remove special symbols like " $\sharp$ , \*, !", etc. and stop words like "a, of, the", etc. Afterwards, we calculate the TF-IDF values of words in each knowledge base.

**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 matching scores of all instance pairs. In our blocking method, we take the predicate as well as top 10 words of the object (ordering by tf-idf values in the knowledge base) as index keys of 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 number of similarity comparisons and improve the efficiency.

**Similarity over Predicates**: The similarity function varies with different predicates. For example, we can use indicator function for the predicate of birthdate, when the value are the same, the indicator is 1, otherwise, 0. For the predicate of comments, we compute cosine similarity based on the tf-idf vectors. In system configuration, we can specify a similarity function for each aligned predicate.

**Similarity Aggregation**: For each instance pair, after acquiring similarity values in terms of multiple aligned predicates, we need to aggregate the similarities to get final matching score. AVG aggregates the similarities by computing the average value [5]. SIGMOID(SIG) aggregates the similarities by computing the average similarities transformed by a sigmoid function [5]. 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:

$$ExpAgg(S) = \frac{\sum_{s_i \in S} w'_i * exp(w''_i * s_i)}{\sum_{s_i \in S} w'_i * exp(w''_i * 1)}$$
(1)

Among them,  $s_i$  is the similarity score in terms of the  $i^{th}$  aligned predicate. We set the weights of the predict "label" and the other predicts as as 16 and 1, respectively.

**Score Matching**: In this task, we don't use the modules of "Unique Subject Matching" and "One-left Object Matching". Each time we choose the pair with the highest score as the aligned pair, we will then update the matching score of each instance pair in the prior queue. With the greedy algorithm of extracting only the most matching pair every time, we control error propagation to some extent. As we can not guarantee a global optimization with the greedy algorithm, we add the later process of validation.

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*, and we also find some rules are very useful. We add validation module to correct some mistakes by some useful rules. In this track, we find that if two instances both contain the predict of "lable", their "label" predicts shall share at least a same token.

#### 1.3 Link to the system and parameters file

The RiMOM-IM system can be found at http://keg.cs.tsinghua.edu.cn/ project/RiMOM/.

## 2 Results

The IM@2014 track contains two subtasks. we present the results and related analysis for the two subtasks in the following subsections.

#### 2.1 Identity Recognition sub-task

The goal of the Identity Recognition sub-task is to determine whether two OWL instances refer to the same real-object. Due to a lack of training data, it is very difficult for us to tuning our parameters. First, we use the default setup to get a preliminary result, and then we check the information of some aligned pairs. We find out that the predict of "label" is very important, so we increase the weight of the "label" predict. Finally, we get 1103 instance pairs as matching ones.

As show in figure 2, the results for the identity task are: Precision 0.65, Recall 0.49, Fmeasure 0.56, which is much lower than we expected. But we are pretty sure that if we have some training set, we can tuning a much better result.

#### 2.2 Similarity Recognition sub-task

The goal of the Similarity Recognition sub-task is to determine the degree of similarity between two OWL instances, even when the two instances describe different realobjects. In our system, we use the traditional cosine similarity measurement, however, if one predicate have many similarity values, we use the maximum value. So in summary, we use maxpooling+cosine similarity. We can find that if two instances describe the same real-objects, their similarity value will usually be larger than that of two instances



Fig. 2. Results of the Identity Recognition Task

which describe different real-objects. Therefore, it's reliable to use the similarity value as a measure to judge whether two instances describe the same real-objects. And we find that if two instances describe different real-objects while have a high similarity value, then their labels are usually different. We can use these two observations for instance matching. As show in figure 3, this similarity strategy is much close to the crowdsourc-



Fig. 3. The Result of Similarity Recognition Task

ing activities'. We have chosen other complexity similarity measurements, but it turns out that this simple measurements works better.

#### 2.3 Discussions on the way to improve the proposed system

Our system need the aligned predicates to select the candidate instance pairs. Our system will use the aligned instance pairs to calculate the similarity values of other instance pairs, which will also need the information of aligned predicates. We need to invent an algorithm to automatically align the predicates. Although there are some algorithms that can align the instances by measuring the similarity values of predicates, none of them use the aligned instance pairs to help to update the similarity values for predicates. We will develop an algorithm that do not need any aligned predicates, but can iteratively use the aligned instance pairs to align the predicates, which will in turn advance the instance alignment.

#### 2.4 Comments on the OAEI 2014 measures

This task is a cross-lingual instance matching task. We find out that we can significantly improve the result by using translation method. And we find that the blocking method also improves the precision of the result. Because the "datatype" of the "object" is always "String", we do not have any relations between any two instances. So we can't use the relation information to improve the recall. This year we use ten keywords for every predict to get more candidate pairs to ensure a high recall.

## **3** Conclusion and future work

In this paper, we present the system of RiMOM-IM in OAEI 2014 Campaign. We participate in one track this year. We described specific techniques we used during this campaign. In our project, we design a new framework to do the instance matching task. Our method effective and efficient.

For now, we need to tune the parameter manually, we will improve it by making the tuning process automatic in the future work.

## References

- Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia - A crystallization point for the web of data. J. Web Sem. 7(3) (2009) 154–165
- Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: YAGO2: A spatially and temporally enhanced knowledge base from wikipedia. Artif. Intell. 194 (2013) 28–61
- Wang, Z., Li, J., Wang, Z., Li, S., Li, M., Zhang, D., Shi, Y., Liu, Y., Zhang, P., Tang, J.: Xlore: A large-scale english-chinese bilingual knowledge graph. In: Proceedings of the ISWC 2013 Posters & Demonstrations Track, Sydney, Australia, October 23, 2013. (2013) 121–124
- Li, J., Tang, J., Li, Y., Luo, Q.: Rimom: A dynamic multistrategy ontology alignment framework. IEEE Trans. Knowl. Data Eng. 21(8) (2009) 1218–1232
- Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology matching with semantic verification. J. Web Sem. 7(3) (2009) 235–251