View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by CiteSeerX

RiMOM Results for OAEI 2015

Yan Zhang, Juanzi Li

Tsinghua University, Beijing, China.
z-y14@mails.tsinghua.edu.cn ljz@ keg.tsinghua.edu.cn

Abstract. This paper presents the results of RiMOM in the Ontology Alignment Evaluation Initiative (OAEI) 2015. We only participated in Instance Matching@OAEI2015. We first describe the overall framework of our matching System (RiMOM); 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.

1 Presentation of the system

As the infrastructure of the Semantic Web, knowledge base has become a dominant mechanism to represent the data semantics on the Web. In this circumstance, a large number of ontological knowledge bases have been built and published, such as DBpedia[1]. , YAGO [2], Xlore [3], etc. In real environment of the Semantic Web, data is always distributed on heterogeneous data sources (ontology). It is inevitable that the knowledge about the same real-world entity may be stored in different knowledge bases. Therefore, there is a growing need to align different knowledge bases so that we can easily get complete information that we are interested in.

Some good results have been achieved in the field of ontology matching [4]. Previous researches always focus on aligning the schema elements (i.e. concepts and properties) in knowledge bases. Most recently, with the rapid development of semantic web, there have been many large-scale ontologies which contain millions of entities. It is obviously that the number of instances is much larger than other elements (e.g. concepts and properties) in these ontologies. For example, the DBpedia contains 882,000 instances of 6 main concepts. Thus, the large-scale instance matching has become the key point in the ontology matching system.

Different from the schema matching, the instance matching always has the following characteristics:

1. The number of instances may be enormous.
2. The schema is straightforward.
3. In practice, the knowledge base is always updated dynamically.

In consideration of these differences, we proposed a large-scale instance matching system, RiMOM.

There are two major techniques in our system, inverted index and multi-strategy:

1. We index the instances based on their objects in two knowledge bases respectively, and then select the instances which contain the same keys as candidate instance

pairs. We limit the number of pairs to be compared by this step, which significantly improve the efficiency of the system.

2. We implement several matchers in our instance matching system, we can execute these matchers in parallel and then aggregate the result according to the characteristics of the source ontologies.

In order to solve the challenges in large-scale instance matching, we propose an instance matching framework RiMOM-2015 (RiMOM-Instance Matching), which is based on our former ontology matching system RiMOM [5]. The RiMOM-2015 framework is designed for large-scale instance matching task specially. It presents a novel multi-strategy method to be fit for different kind of ontology and employs inverted index to imporve the efficiency.

1.1 State, purpose, general statement

This section describes the overall framework of RiMOM. The overview of the instance matching system is shown in Fig. 1. The system includes seven modules, i.e., *Preprocess*, *Predicate Alignment*, *Mathcher Choosing*, *Candidate Pair Generation*, *Matching Score Calculation*, *Instance Alignment* and *Validation*. The sequences of the process are shown in the Fig. 1. We illustrate the process as follows.

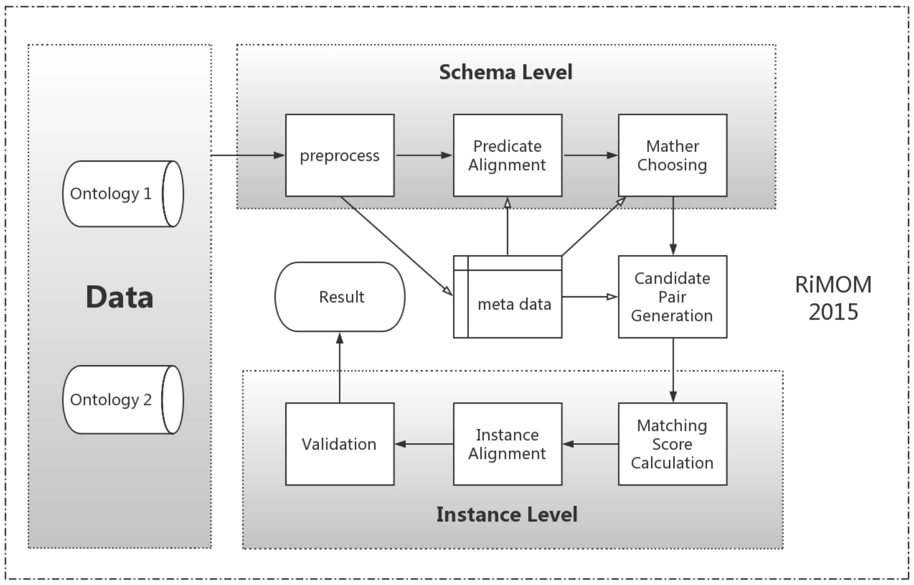

Fig. 1. Framework of RiMOM 2015

1. **Preprocess:** The system begins with *Preprocess*, which loads the ontologies and parameters into system. In the meantime, preprocessor can get some meta data about the two ontologies, which will be used in the later processes, *Predicate alignment* and *Matcher choosing*
2. **Predicate Alignment:** In this process, we will get the alignments of the predicates between the two ontologies. Currently, in our system, this process is semi-automatic.
3. **Matcher choosing:** The system will choose the most suitable one or more matchers according to the meta data of the ontologies.
4. **Candidate Pairs Generation:** In this step, we get the candidate pair when the instances have the same literal objects on some discriminatory predicate.
5. **Matching Score Calculation:** After the candidate set generation, we calculate more accurate similarity using the algorithm chosen by step 3. In this task, the vector distance similarity was calculated between each candidate pair.
6. **Instance Alignment:** According to the similarity calculated in step 5, we get the final instance alignment.
7. **Validation:** We will evaluate the alignment result on Precision, Recall and F1-Measure if there is validation data set.

1.2 Specific techniques used

This year we only participate in the Instance Matching track. We will describe specific techniques used in this track.

Data Preprocessing: First, we remove some stop words like ”a, of, the”, etc. Afterwards, we calculate the TF-IDF values of words in each knowledge base. We also calculate some information of each predicate, in order to find the important predicates.

Predicate Alignment: It is apparent that we should get the alignment of the predicates before we calculate the similarity of instances. The predicates can express rich semantics, and there exists one-to-one, one-to-many, or many-to-many relationships among these predicates. We can find some of one-to-one relationships through calculating the *Jaccard Similarity* of the two predicates. i.e.

$$sim(p_i, p_j) = \frac{|O_{p_i} \cap O_{p_j}|}{|O_{p_i} \cup O_{p_j}|}$$

where p_i and p_j are predicates in two ontologies respectively. O_{p_j} is the $range$ of the predicate p_j.

There are also some one-to-many relationships. We get the alignments of them by manual regulations, e.g.

$$object(p_i) = \sum_{j=1}^{n} object(p_j)$$

$$object(p_i) = \max_{j=1..n} object(p_j)$$

$$object(p_i) = \min_{j=1..n} object(p_j)$$

Candidate Pairs Generation: This step 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 method, we firstly generate the inverted index on the objects. instance pairs are selected into the candidate set when they have common objects. This method may reduce the recall, but it also reduce the scale of computation significantly.

Multi-Strategy: We implement several matchers in our system, e.g. label-based approach and structure-based approach. In the preprocess step, we will compare the schema of the two ontologies. If the range of predicates is similar, the label-based approach will play a key role in the matching process. Otherwise, the literal properties are not similar (e.g. the two ontologies are defined in different languages), label-based approach will not be effective. In this case, we will get some supplementary information (e.g. machine translation, WordNet), or use structure-based appraoch.

Similarity Calculation: In OAEI 2015 instance matching track, the ontologies are all defined in the same language, English. In the tasks which we took part in, $author-dis$ and $author-rec$, the schema of the ontologies tend to be similar. So label-based vector distance matcher is chosen to calculate the similarity of the instances, it is defined as follows:

$$L_a = Objects(I_a)$$

where I_a is an instance, L_a is a list which contains all of the objects of the instance I_a.

$$Sim(I_a, I_b) = Sim(L_a, L_b) = \frac{1}{|L_a|} \sum_{O_a \in L_a} \max(Sim(O_a, O_b) | O_b \in L_b)$$

where O_a is one of the objects in the list L_a. We define the similarity of the two instances equals to the similarity of their objects list. For each O_a in L_a, we find a most similar object O_b in L_b. The algorithm varies with the data type of the object. For example, for date, we use the indicator function. The indicator function will be 1 when the dates are the same, otherwise, 0. For some literal properties, such as ”title”, we compute cosine similarity based on the tf-idf vectors.

Instance Alignment After we get the accurate similarity, for each instance in source ontology, we choose the instance which has the best score in target ontology. Then we filter the result on a certain threshold and get the final Instance Alignment.

1.3 Link to the system and parameters file

The RiMOM system (2015 version) can be found at `https://www.dropbox.com/s/6bx4pb46ytvddvy/RiMOM.zip?oref=e`.

2 Results

The Instance Matching track contains five subtasks. we present the results and related analysis for the two subtasks (author-disambiguation and author-recognition) in the following subsections.

2.1 Author Disambiguation sub-task

The goal of the author-dis task is to link OWL instances referring to the same person (i.e., author) based on their publications. We can use the Sandbox (small scale data set) to tune our parameters. The class 'author' have only one literal properties, 'name'. So we must get alignments on the class 'publication'. Finally, we get 854 pairs for Sandbox task, and 8428 pairs for Mainbox task.

	Expected mappings	Retrieved mappings	Precision	Recall	F-measure
EXONA	854	854	0.941	0.941	0.941
InsMT+	854	722	0.834	0.705	0.764
Lily	854	854	0.981	0.981	0.981
LogMap	854	779	0.994	0.906	0.948
RiMOM	854	854	0.929	0.929	0.929

Table 1. The result for Author-dis sandbox

	Expected mappings	Retrieved mappings	Precision	Recall	F-measure
EXONA	8428	144827	0	0	NaN
InsMT+	8428	7372	0.76	0.665	0.709
Lily	8428	8428	0.964	0.964	0.964
LogMap	8428	7030	0.996	0.831	0.906
RiMOM	8428	8428	0.911	0.911	0.911

Table 2. The result for Author-dis mainbox

The reference alignments of sandbox are provided by sponsor, so we only pay attention to mainbox. As shown in table 2, the results for the author-dis mainbox task are: Precision 0.911, Recall 0.911, F-measure 0.911, which is slightly lower than sandbox. Afterwards, we find that the property 'title' plays a key role in publication. So we think that we can get a better result if we do some deeper work on it.

2.2 Author Recognition sub-task

The goal of the Author-rec task is to associate a person (i.e., author) with the corresponding publication report containing aggregated information about the publication activity of the person, such as number of publications, h-index, years of activity, number of citations. The final goal is similar with the Author-dis task, but there are some changes on schema of the ontology. The most remarkable is that there exists one-to-many relationships between the properties. So we add some manual regulation to solve the problem.

As show in table 4, RiMOM get a excellent result on author-rec task. The results for the author-dis mainbox task are: Precision 0.999, Recall 0.999, Fmeasure 0.999, which expresses that the algorithm we implement is very suitable for this task.

	Expected mappings	Retrieved mappings	Precision	Recall	F-measure
EXONA	854	854	0.518	0.518	0.518
InsMT+	854	90	0.556	0.059	0.106
Lily	854	854	1.0	1.0	1.0
LogMap	854	854	1.0	1.0	1.0
RiMOM	854	854	1.0	1.0	1.0

Table 3. The result for Author-rec sandbox

	Expected mappings	Retrieved mappings	Precision	Recall	F-measure
EXONA	8428	8428	0.409	0.409	0.409
InsMT+	8428	961	0.246	0.028	0.05
Lily	8428	8424	0.999	0.998	0.999
LogMap	8428	8436	0.999	1.0	0.999
RiMOM	8428	8428	0.999	0.999	0.999

Table 4. The result for Author-rec mainbox

2.3 Discussions on the way to improve the proposed system

Our system need align the predicates before instance matching, and in this process, the system is required to scan all of the instances in the ontology, which may cause a waste of time. In addition, the process of $PredicateAlignment$ is semi-automatic, we have to add some manual regulations to deal with the one-to-many relationships.

In conclusion, we hope to develop our system through inventing an algorithm to align the predicates automatically and iteratively. Firstly we can use the values of predicates to align the instances, and in turn, the aligned instances will help us to update the similarity for predicates. In this way, we will gradually get the final alignment result.

2.4 Comments on the OAEI 2015 measures

These two tasks are instance matching task on publication data set. We use the reference of the sandbox to tune the parameters,and it turns out that our approach is effective. We also find that the inverted index not only improve efficiency, but reduce the mistake and increase the Precision. There are also some aspects we are not satisfied with. For time's sake, we don't take part in other three tasks. Finally, we are looking forward to making some progress in the next OAEI campaign.

3 Conclusion and future work

In this paper, we present the system of RiMOM in OAEI 2015 Campaign. We participate in intance matching track this year. We described specific techniques we used in the task. In our project, we design a new framework to deal with the instance matching task. The result turns out that our method is effective and efficient.

In the future, we will develop an iterative algorithm to align the predicates automatically.

References

1. 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
2. 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
3. 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
4. Shvaiko, P., Euzenat, J.: Ontology matching: State of the art and future challenges. IEEE Trans. Knowl. Data Eng. **25**(1) (2013) 158–176
5. 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