RiMOM Results for OAEI 2010

Zhichun Wang[1], Xiao Zhang[1], Lei Hou[1],
Yue Zhao[2], Juanzi Li[1], Yu Qi[3] , Jie Tang[1]

[1]Tsinghua University, Beijing, China
{zcwang,zhangxiao,greener,ljz,tangjie}@keg.cs.tsinghua.edu.cn
[2]Beihang University, Beijing, China
zhaoy1030@gmail.com
[3]National University of Defense Technology, Changsha, China
qiyu_418@sina.com

Abstract. This paper presents the results of RiMOM in the Ontology Alignment Evaluation Initiative (OAEI) 2010. We participate in three tracks of the campaign: Benchmark, IM@OAEI2010 (IMEI), and Very Large Crosslingual Resources (VLCR). We first describe the basic alignment process and alignment strategies in RiMOM, and then we present specific techniques used for different tracks. At last we give some comments on our results and discuss some future work on RiMOM.

1 Presentation of the system

Recently, ontology alignment has been developed as a key technology to solve interoperability problems across heterogonous data sources. Many automatic ontology alignment systems have been proposed and achieve good performance in real world data. With the development of Linked Data [1] and various social network websites, huge amount of semantic data is published on the web, which not only poses new challenges over traditional schema level ontology alignment algorithms, but also demands new techniques for instance matching.

RiMOM is a multistrategy dynamic ontology alignment system [2]. It implements several different matching strategies which are defined based on different ontological information. For each individual matching task, RiMOM can automatically and dynamically combine multiple strategies to generate a composed matching result. Recently, some new features were added into the new version of RiMOM which enable it to deal with unbalanced ontology matching [3], user interactive ontology matching [4], and large scale instance matching.

1.1 State, purpose, general statement

Currently, RiMOM is developed with a flexible framework for ontology alignment, where different kinds of alignment strategies can be plugged and configured easily. Fig 1 shows the architecture of RiMOM system.

The whole system consists of three layers: interface layer, task layer and

component layer. In the interface layer, RiMOM provides a graphical user interface to allow users to customize the matching procedure: including selecting preferred components, setting the parameters for the system, etc. In semi-automatic ontology matching, user can also get involved in the matching process via the user interface. The task layer stores parameters of the alignment tasks, and controls the execution process of components in the component layer. In component layer, we define five groups of executable components, including preprocessor, matcher, aggregator, postprocessor and evaluator. In each group, there are several instantiated components. For a certain alignment task, user can select appropriate components and execute them in desired sequence.

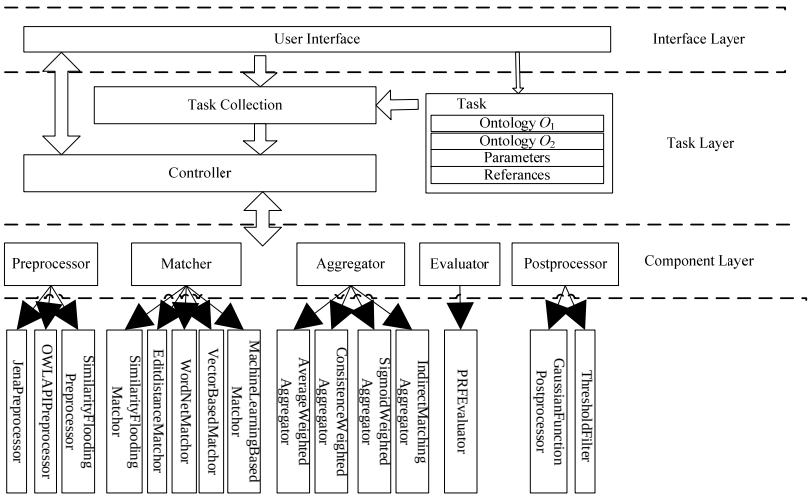

Fig 1. Architecture of RiMOM system

1.2 Specific techniques used

This year we participate in three tracks of the campaign: Benchmark, IM@OAEI2010 (IMEI), and Very Large Crosslingual Resources (VLCR). We describe specific techniques used in different tracks as follows:

Benchmark Track

For benchmark track, we use three matching strategies:

(1) Name based strategy: In this strategy, we calculate the edit distance between labels of two entities. Edit distance estimates the number of operation needed to convert one string into another. We define $(1-\#op/max\_length(l_1,l_2))$ as the similarity of two labels, where $\#op$ indicates the number of operations, $max\_length(l_1,l_2)$ represents the maximal length of the two labels.

(2) Metadata based strategy: In this strategy, we treat the information of each entity as a document, which consists of words in entity's label and comment. Then we construct a weighted feature vector using tf-idf technology, the similarity between two entities is then calculated as the cosine of the two vectors.

(3) Instance based strategy: In this strategy, we also construct a document for each entity, but the words are from the instances related to that entity. For a class entity, words in the label, comment and property value of all its instances are extracted as the entity's document; for a property entity, all the values it occurs in instances are extracted as the entity's document. Then the similarity between two entities is calculated as in Metadata based strategy.

When combining the results of different matching strategies, we use a different method from which we used in OAEI 2008 and 2009. Instead of aggregating similarity values before extracting final alignment, we first extract alignment based on each individual strategy by threshold filtering method, and then combine alignments of different strategies together. A similarity propagation procedure based on structure information is performed to find more mappings. The similarity propagation procedure is implemented in iteration; in each iteration, the similarity is propagated from already found mappings to the rest candidate mappings, candidate mappings which get high similarity are then added to found mappings; this process is repeated until no more mapping is found. This combination method can generate alignments with very high precision with acceptable recall.

Data Interlinking track

The DI (Data Interlinking) track is designed to test the ontology matching systems' ability on link generation of LinkedData. There are five datasets, i.e. DailyMed, Diseasome, DrugBank, Sider and LinkedMDB, requested to be matched to related datasets in the LinkedData respectively. These data sets are all comes from the real world data and in relatively larger scale than the generated dataset. We choose four datasets in the domain of medicine to test our algorithm while exclude the linkedMDB dataset. According to our observations on the instance data, we split the information in the instance into six categories: the URL, the Meta Information, the Name, the string type information, the non-string type information and the neighboring information. Among the six categories the Name, which usually comes from the rdfs:label property or other ontology specific property such as foaf:name) is the most distinguishing feature to identify an instance. In addition, the natural language information and the neighboring instances are very useful, too. Thus we propose a vector based method for the DI track. We build two vectors referred to as Name Vector and Virtual Document for each instance. The Name Vector is constructed by accumulating the terms in the Name property values and setting the occurrence of each term as its weight. For Virtual Document, we first collect the terms of the each instance's descriptions and annotations then fetch the local information of its neighboring instances to construct a comprehensive vector. Because the Virtual Document Space is much larger, we compute the tf-idf value of each term as its weight. The similarity between two instances is calculated as the weighted sum of their similarity (Cosine Distance) on two kinds of vectors respectively. However, this method is infeasible on large scale input because pair-wise comparisons on instances are too costly. Thus we introduce a candidate selection process. Only the instance pairs which are selected as candidate mappings are compared. Generally we use two rules for candidate selection: 1) instances with common terms in their Name Vectors; 2) instances with common top weighted terms in their Virtual Documents. To utilize the functionality, we build inverted index of instances for terms in Name Vector and top weighted terms in the Virtual Document. Consequently our algorithm

can generate the candidates very quickly and eliminate the meaningless comparisons between unrelated instances. Several experiment results show that the candidate selection will not eliminate the possible alignments in most of the cases. In the following phase of the algorithm, we may use the Meta Information and non-string type values as restrictions to filter the results according to the instance characteristics. For example, a common one is that those instances whose classes are not matched will be filtered out. At last a threshold is used on similarity for the final result. Totally speaking, this method is a generic and efficient method for instance matching.

IIMB and PR track

Traditionally, information of individuals in an ontology is frequently utilized in supporting of schema matching. Inversely, information of schema is of equal importance in alignment of individuals that are sharing the same ontology structure. Thus, for the Instance Matching Track of this year, we take more about schema information, especially classes and properties, into consideration in aligning individuals.

For Instance Matching, our main idea is that we classify individuals by their classes, complete information of each individual as complete as possible, run matching algorithm for each class respectively, and compute similarity of two candidates based on weight-mean of properties assigned with specified weights. And the algorithm can be generalized as four consecutive phase:

Preprocessing: Read and store the schema information for further use. Build a local schema that connects properties and classes and implement it by learning information of individuals.

Information Complementation: Modify the information of each individual, aiming at making them as complete as possible. We defined some rules for judging the validity of values, as well as for solving the transformations in value, structure and logical. Reclassify individuals by recognizing and comparing properties they carry with those in classes, based on our local schema implemented in the previous phase.

Matching: Given the facts that different properties of individuals play quite different roles, and that every individual has its unique characteristic(s), for each property, we assign it with a specified weight and combine this weight with string-based similarity value computed under Edit Distance or Vector based algorithm. We assign the weight-mean of properties as the final similarity value.

Spread Similarity: In order to fully utilize the connection of individuals, we apply a similarity-flooding-like algorithm to spread the similarity.

1.3 Adaptations made for the evaluation

In order to deal with large scale data sets, we use an inverted index technique to accelerate the speed of locating and reading data.

1.4 Link to the system and parameters file

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

1.5 Link to the set of provided alignments (in align format)

The results of RiMOM for OAEI 2010 Campaign are available at http://keg.cs.tsinghua.edu.cn/project/RiMOM/OAEI2010/

2 Results

As introduced above, RiMOM participates in three tracks in OAEI 2010; we present the results and related analysis below.

2.1 Benchmark

There are 111 alignment tasks in benchmark data set; we divide these tasks into three groups: 1xx, 2xx, and 3xx. We compare the results of RiMOM in OAEI 2010 and OAEI 2009 [5] in Table 1. It can be observed that the performance of RiMOM in 1xx task continues to be perfect as last year; as for the 2xx task, the result of this year is better than that of last year, with regard to both precision and recall; the precision of 3xx increases this year, but the recall decreases, while the F1-measure is almost the same as last year. Overall, the precision, recall and F1-measure for the entire benchmark data set of RiMOM this year achieve 99% precision, 84% recall and an F1-measure of 91%. Compared with last year's result, there are 6% improvement on precision, 2% improvement on recall and 4% improvement on F1-measure.

Table 1. Benchmark test results of RiMOM in OAEI 2010 and OAEI 2009
(Values are real precision and recall and not an average of precision and recall)

Test	OAEI 2010			OAEI 2009		
	Prec.	Rec.	F1	Prec.	Rec.	F1
1xx	1.00	1.00	1.00	1.00	1.00	1.00
2xx	0.99	0.83	0.91	0.93	0.81	0.87
3xx	0.91	0.74	0.82	0.81	0.82	0.81
H-mean	0.99	0.84	0.91	0.93	0.82	0.87

2.2 DI track of IM@OAEI2010

We generate results for four of five datasets in the track except the LinkedMDB dataset. Since we are requested to mapping each dataset to several related datasets in LinkedData and these datasets not provided in the track, we download these datasets and transfer them into RDF format using Jena. As a result we cannot get some datasets such as STITCH because there is only a SPARQL endpoint for it. We also found there are many duplicate entries in the reference alignment of Sider and the namespace for DBpedia in the reference alignment of Drugbank is not uniform, we adjust these reference files to get the final result of our algorithm. We set the parameter of our algorithm as NameWeight = 0.6 and threshold = 0.55. The result of

Sider dataset is shown in Table 2. From the result we can see that according to the different characteristics of the instance file, the results may be very different: some are high in precision and some are high in recall. For those high in recall but low in precision, more careful filter may be added to the algorithm by studying the data. On the other hand, for those low in recall, the threshold may be cut down.

Table 2. The result of Sider Dataset

DataSet	DBpedia	DailyMed	Diseasome	Drugbank	TCM	STITCH	TOTAL
Precision	0.717	0.567	0.315	0.961	0.778	/	0.617
Recall	0.482	0.706	0.837	0.342	0.812	/	0.467
F-Measure	0.576	0.629	0.458	0.504	0.795	/	0.532

The result of DailyMed dataset is shown in Table 3. The result of our algorithm is extremely bad in the LinkedCT dataset. It generates a lot of results (up to 100,000) so that the precision is very low. Because of the dominance of LinkedCT results in the reference, our result in total is not good, too. According to our observation on the reference alignment of LinkedCT, they are automatically generated from the owl:SeeAlso property in the file. After reviewing some of our results, we found that many of our results are reasonable but some of the references are not, we think the reference alignment is not very complete and sound. However, our algorithm cannot generate good results from DBpedia means we need much more improvement on it. The other two datasets with LinkedCT reference, Diseasome and DrugBank are similar in results.

Table 3. The result of DailyMed dataset

DataSet	DBpedia	LinkedCT	TCM	Sider	TOTAL
Precision	0.246	0.070	0.159	0.567	0.085
Recall	0.293	0.235	0.535	0.706	0.296
F-Measure	0.267	0.107	0.123	0.629	0.132

Table 4. Results of IIMB

IIMB_SMALL				**IIMB_LARGE**			
Dataset	**Prec.**	**Rec.**	**F1**	**Dataset**	**Prec.**	**Rec.**	**F1**
001 - 020	0.975	0.975	0.975	**001 - 020**	0.997	0.994	0.995
021 - 030	0.861	0.710	0.778	**021 - 030**	0.798	0.696	0.744
031 - 060	0.913	0.953	0.933	**031 - 040**	0.843	0.766	0.803
061 - 070	0.809	0.639	0.714	**041 - 060**	0.877	0.976	0.924
071 - 080	0.792	0.500	0.613	**061 - 070**	0.663	0.586	0.622
				071 - 080	0.575	0.557	0.566

2.3 IIMB track of IM@OAEI2010

The result for IIMB_SMALL and IIMB_LARGE is shown in Table 4. As the number of datasets increases, the text-based information the dataset contains decrease while complex combination of modifications increase, thus the performance of our algorithm decreases since it is anyway fundamentally based on string comparison. We can also see that with the amount of instances grows, the influences brought by the noise increase, which do nothing but harm to effect of our algorithm.

2.4 PR track of IM@OAEI2010

PR track consists of three subtasks; the results for these tasks are shown in Table 5. It can be observed that RiMOM gets perfect performance on the first task; for the second task, RiMOM gets really good recall and the precision is 95.2%; for the last task, the precision and recall both decrease compared to the former two tasks.

Table 5. Results of PR

Dataset	Precision	Recall	F-Measure
Person11 - Person12	1.0	1.0	1.000
Person11 - Person12	0.952	0.99	0.971
Restaurant1 - Restaurant2	0.86	0.768	0.811

2.5 VLCR track

The purpose of VLCR task is to match three resources to each other, namely, the Thesaurus of the Netherlands Institute for Sound and Vision (called GTAA), the New York Times subject headings and DBpedia. Each resource consists of lots of instances: 142,000 in GTAA, 12,000 in NYT and 7,500,000 in DBpedia. Table 6 lists the number of the mapping we found.

Table 6. Result for VLCR task

Dataset		Number of mappings
NYT-DBpedia		9257
GTAA-DBpedia		68337
NYT-GTAA	Direct mapping	4324
	Indirect mapping	4487

Due to the lack of information, sometimes it is very difficult to match two instances in NYT and GTAA directly. Since we have mapped the two relatively small instance sets to DBpedia, it is possible to use the map results to get more maps between the two small one. Instances in NYT and GTAA matches to the same instance in DBpedia will be added to the final results.As shown in the table, NYT–DBpedia, GTAA-DBpedia and NYT-GTAA are three subtasks of VLCR task. Indirect matching find 163(rise by 3.7%) new mappings in NYT-GTAA task.

3 General comments

By far instance matching, especially matching on real world instance is still a very challenging problem. Instance Matching is of great importance for bringing the ontology matching into practical use with its wide range of application scenarios. Instance matching shows its special characteristics compared with the conventional schema matching and the large scale nature of instance matching is a big obstacle to employ the existing methods. A relatively generic and efficient method for instance matching is in great need. The IMEI track of OAEI 2010 provides a good platform to test the instance matching algorithms and this area will attract more attention in the community.

4 Conclusion

In this paper, we present the results of RiMOM in OAEI 2010 Campaign. We participate in three tracks this year, including Benchmark, IMEI, and VLCR. We have presented the architecture of RiMOM system and described specific techniques used in this campaign. In this campaign, we design a new strategy combination method for benchmark tracks, and get better performance than last year. We particularly focus on the instance matching task; propose some new strategies for these tasks. The results illustrates that our system RiMOM can achieve good performance in both schema matching and instance matching tracks.

Acknowledgement:

The work is supported by the National Natural Science Foundation of China (No. 60973102), the National Basic Research Program of China (973 Program) (No. 2007CB310803), the National High-tech R&D Program (No. 2009AA01Z138), it is also supported by IBM SUR joint project.

References

1. http://linkeddata.org/.
2. J. Li, J. Tang, Y. Li, and Q. Luo. RiMOM: A dynamic multi-strategy ontology alignment framework. *IEEE Transaction on Knowledge and Data Engineering*, 21(8):1218–1232, Aug 2009.
3. Q. Zhong, H. Li, J. Li, G. Xie, and J. Tang. A Gauss Function based approach for unbalanced ontology matching. In *Proc. of the 2009 ACM SIGMOD international conference on Management of data (SIGMOD'2009)*, Jul 2009.
4. F. Shi, J. Li, and J. Tang. Actively learning ontology matching via user interaction. In *Proc. of the 8th International Conference of Semantic Web (ISWC'2009)*, Oct 2009.
5. X. Zhang, Q. Zhong, J. Li, J. Tang, G. Xie, and H. Li. RiMOM results for OAEI 2008. In *Proc. of the Third International Workshop on Ontology Matching (OM'08)*, 2008.