LiSTOMS: a Light-weighted Self-tuning Ontology Mapping System

Zhen Zhen Junyi Shen
Institute of Computer Software,
Xi'an Jiaotong University,
Xi'an, 710049, China
E-mail: zhenzh@stu.xjtu.edu.cn

Jinwei Zhao Jianjun Qian
China Defense Science and Technology
Information Center,
Beijing, 100142, China

Abstract

We argue that it is more practical to address the ontology mapping self-tuning problem in a whole system context instead of in a single matcher context. In this paper we introduce RMOMS, a Reference Model for Ontology Mapping Systems, consisting of six parts, the Preprocessor, the Dispatcher, the Matcher(s), the Aggregator, the Pruner, and the User Interface, with which to disassemble the self-tuning problem into more feasible units. We propose Maximum Weight Bipartite Graph Matching method for self-tuning matchers and Stable Match method for self-tuning aggregator, and test them in LiSTOMS, a light-weighted prototype sample of RMOMS. With comparison with some notable systems, LiSTOMS shows leading recall rate and competing precision rate.

1. Introduction

The final goal for ontology mapping is to find the correspondences between semantically related entities of different ontologies correctly, completely, and efficiently. Since an algorithm might be effective only for a few scenarios, and not for others, most ontology mapping systems adopt multiple matching methods, or matchers, to improve generality of mapping [1], finding linguistic and/or structural similarities between ontologies entities. However, these systems have to use various parameters for better fitting in with the need of diverse context, and the matching outcome of these systems heavily rely on their parameters, even within the same scenario, changing the parameters can lead to significant different outcome. So the more we rely on the trick of setting weights and thresholds manually, the less usability such systems may retain, especially in a dynamic application context, such as the Web. Tuning ontology mapping system automatically has become one of the challenges in ontology mapping field [2].

To disassemble this problem into more feasible units, we introduce RMOMS, a Reference Model for Ontology Mapping Systems, which consists of six parts, the *Preprocessor*, the *Dispatcher*, the *Matcher(s)*, the *Aggregator*, the *Pruner*, and the *User interface*. Within this framework, the solution to the problem lies largely in four parts: the Dispatcher, the Matcher(s), the Aggregator and the Pruner. We concentrate on the Matchers and the Aggregator in this paper.

The remainder of this paper organized as follows. In Section 2, we brief the related works. In Section 3, we present RMOMS and our sample implementation of this model, LiSTOMS, a Light-weighted Self-Tuning Ontology Mapping System. In Section 4 and 5, we propose and test *Maximum Weight Bipartite Graph Matching (MWBGM)* method for self-tuning matchers and *Stable Match* method for self-tuning aggregator used in LiSTOMS. Finally, in Section 6 we summarize our findings of this paper.

2. Related Works

As stated in [3], tuning a multiple components ontology mapping system is a must, but skill- and time-intensive process, and efforts have been made to reduce the cost.

eTuner project [3][4] proposes a staged tuning method, where the matchers are organized in an execution tree to avoid exhaustive tuning. Output of the lower level matchers serves as input to the higher level matchers, and tuning starts with the matchers at the leaf level, then moves to the matchers at the upper level, and so on.

Another approach to parameter tuning is machine learning. For example, LSD algorithm [5] performs a liner regression to determine the weights of learners, and then uses weighted average to combine learners'

978-0-7695-4191-4/10 $26.00 © 2010 IEEE
DOI 10.1109/WI-IAT.2010.173

grades. Another example is APFEL [6], which uses users' feedback to generate new hypotheses for the tuning process.

We also investigated some of the systems which took part in the last few Ontology Alignment Evaluation Initiative (OAEI) campaigns, with special focus on their architecture, components and tuning methods.

Falcon-AO [7] uses four matchers, and the association between detected similarities and matchers to be combined is predefined. For example, if the two ontologies have a high linguistic factor, Falcon-AO will reduce the thresholds of its linguistic matchers and leak more output from these matchers.

RiMOM [8] implements more than eight matchers. Similarly to Falcon-AO, it holds a predefined association its strategy with three ontology feature factors: *label similarity, structure similarity, and label meaning*. For example, if the two ontologies have a high structure similarity factor, the system will use similarity-propagation based strategies on them. Multiple matching results are combined with predefined experimental weights.

AgreementMaker [9] features an extensible architecture to incorporate new methods and to tune their performance. It implements a variety of matchers, and allows serial and parallel composition of them. It employs Linear Weighted Combination (LWC) method and bipartite graph based selection method to get the final alignment, where the weights of LWC can be assigned manually or automatically, using a matcher confidence measure.

3. The Reference Model of Ontology Mapping System (RMOMS)

3.1 Components of RMOMS

The direction of matcher self-tuning is still largely unexplored[10], and the problem gets even more complicated since most current mapping systems employs more than one matcher to improve robustness. We argue that it is more practical to consider the tuning problem in a whole system context instead of in a single matcher context.

We get a good and easy view of mainstream ontology mapping systems by checking results of OAEI ontology matching campaigns. Besides Falcon-AO, RiMOM and AgreementMaker mentioned above, we also studied some other notable systems, such as ASMOV [11], COMA++ [12].

Each of these systems contains similar components with those of others, more or less, and there appears to be a generic mapping process, however, they use different term "dialect" to name their components, which makes it hard for further discussion. We present here RMOMS, a Reference Model of Ontology Mapping System, inspired by the systems we studied, as well as AUTOMS-F [13], to help us to anatomize the self-tuning problem within a system context. RMOMS includes following six components:

The Preprocessor imports and parses the ontologies, and probably provides formatted storage of them. Some systems, for example Falcon-AO and COMA++, partition the large ontologies into smaller blocks for later matching operations,.

The *Dispatcher* selects and combines proper matchers to work, in run-time or design-time fashion, serialized or paralleled, or even iteratively, by user interaction or automatic analyzing features of ontology. The Dispatcher is actually the central controller of matching process. Many systems do not have an explicit dispatching module, while COMA++ has two components named Execution Engine and Match Customizer. Matching processing is performed in Execution Engine in the form of match iterations, and each iteration can be configured individually with Match Customizer.

The *Matcher* is the key of the system and controlled by the Dispatcher, implementing matching algorithms. We call the one which computing with single algorithm an atomic matcher, and that computing with multiple algorithms a hybrid matcher. COMA++ extends its previous prototype COMA[14], implementing atomic matchers such as Affix, N-Gram, and EditDistance matcher, and hybrid matchers such as Name matcher, integrating three atomic matchers above, and TypeName matcher further integrating Name and DataType matchers.

The *Aggregator* combines multiple similarity values from matchers into one value. In AgreementMaker, the Aggregator exists as its third layer matchers to obtain a unique matching by combining the results of two or more matchers.

The *Pruner* removes semantically invalid or unsatisfiable correspondences. ASMOV performs comprehensive validation and pruning process in each mapping iteration, with five kinds of semantic verification inferences, such as multiple-entity correspondences, disjointness-equivalence contradiction, subsumption incompleteness, and so on.

The *User interface* gets the user inputs, such as the parameters, weights and thresholds, or assists uses to get final alignment by suggesting plausible matching candidates. All systems investigated by us have graphic user interface.

We summarize in Table 1 the components of RMOMS with those of ASMOV, COMA++, Falcon-AO, and RiMOM.

Table 1. **Components of RMOMS and ontology mapping systems**

RMOMS	ASMOV	COMA++	Falcon-AO	RiMOM
Preprocessor	Have	Have	Have	Have
Dispatcher	Have	Have	Have	Have
Matcher(s)	Have	Have	Have	Have
Aggregator	Have	Have	Have	Have
Pruner	Have	No	No	Have
User Interface	GUI	GUI	GUI	GUI

With RMOMS, it is clarified now that a self-tuning ontology mapping system mostly features its self-tuning dispatcher, matchers, aggregator, and pruner. However, only two of them are addressed in this paper, the matchers and aggregator.

3.2 LiSTOMS: a prototype sample of RMOMS

3.2.1. Overview. We design LiSTOMS as our first prototype and evaluation platform of RMOMS, with only four components developed by now, the preprocessor, six matchers, the aggregator, with a graphic user interface. The system has an extensible matcher architecture so new matchers can be included and used in combination with others in future.

3.2.2. The matchers. We design and implement two sets of matchers, and each set consists of three matchers. The first set includes one atomic matcher and two hybrid matchers. The first matcher, named the EW matcher, which combines an edit distance matcher [15] and a Word-Net matcher using Wu-Palmer algorithm [16]. The second matcher, called the CDW matcher using Stoilos-Stamou-Kollias algorithm [17], and the third matcher, named the CONTX matcher, computes three facet contexts of concepts, which are structure facet, attribute facet and instance facet [18], using the algorithm of vector space model [19]. The other set consists of variants of the above three matchers separately, named the EW+, CDW+, and CONTX+ matcher, optimized with *Maxium Weight Bipartite Graph Matching (MWBGM)* method (presented in Section 4.1), which has been used for correspondences selection by Similarity Flooding [20] matcher, and by AgreementMaker aggregator. We expand its usage as a generic self-tuning add-on for all matchers.

3.3.3. The aggregator. We design the aggregator featuring *Stable Match* method (introduced in Section 5.1) to assign weights for matcher automatically and using linear weighted combination to obtain the weighted average for the similarity values from different matchers. The aggregator is also equipped with the MWBGM method to optimize the final outcome of the system.

4. Maxium Weight Bipartite Graph Matching method for self-tuning matcher

4.1. Overview

We narrowed our study to deal with one-to-one matching case, because this kind of case is often required in real-world scenarios, and many-to-many matching can hopefully be established based on it [21].

Given the source ontology has m elements and the target ontology has n elements, we assume that the matcher produces one-to-one correspondences among entities of different ontologies, and the amount of the correspondences is the minimum of m and n. We also assume that the matcher expects the overall 'best' similarity.

Inspired by the study of Similarity Flooding and AgreementMaker, we transform this problem into an well-known optimization problem, named the *Assignment Problem*, and adopt to model the candidate correspondences as weighted bipartite graph $G= (S \cup T, E)$, where S contains the source ontology entities, T contains the target ontology entities, and E contains one edge from S to T for each correspondence weighted with a similarity value computed by the matcher. Algorithms such as the Hungarian Method [22] and the Shortest Augmenting Path [23] algorithm can be adopted to solve this problem then, finding out a maximum weight matching M. For each vertex in G at most one adjacent edge is contained in M, where the sum of the weights of the edges is maximized.

4.2. Self-tuning matcher test

4.2.1. Test One. We compare the original EW, CDW and CONTX matcher, with their variants, the EW+, CDW+, and CONTX+ matcher separately, using #205 and #101 ontologies of OAEI 2008 benchmark (http://oaei.ontologymatching.org/2008/results/benchmarks). #101 is used as the reference ontology.

For the original matchers, we set the thresholds on purpose to make the amount of output correspondences close to that of correct ones, and then we can get relatively high F-Measure. We also observe that these three original matchers use different thresholds to achieve this. However, the overall results of original matchers are obviously inferior to those of the optimized ones. In real scenarios, we cannot know how

many correct correspondences there are between different ontologies, so we have no means to know what the 'proper' thresholds are, which we can only know in experimental scenarios. When we decrease the thresholds, we can get higher recall rate, at the expense of the precision and the comprehensive F-value. Especially in the case of EW matcher, we lower the threshold from 0.9 to 0.8, the recall rate increases from 0.42 to 0.64 but the precision rate drops from 0.38 to 0.19.

Table 2. **Results of test one**

Matchers	Thresholds	Correct/All	Precision	Recall	F-Measure
EW	0.90	14/37	0.38	0.42	0.40
	0.80	21/111	0.19	0.64	0.29
EW+	-	20/33	0.61	0.61	0.61
CDW	0.55	13/35	0.37	0.39	0.38
	0.40	16/65	0.25	0.48	0.33
CDW+	-	13/33	0.39	0.39	0.39
CONTX	0.60	29/33	0.88	0.88	0.88
	0.40	31/39	0.79	0.94	0.86
CONTX+	-	33/33	1.00	1.00	1.00

4.2.2. Test Two. In this test, we compare EW+, I-Sub+, and CONTX+ matcher with ASMOV, LILY and RiMOM, using #301, #302, #303, #304 and #101 ontologies of OAEI 2008 benchmark suite. #101 is used as the reference ontology as well.

Table 3. **Results of test two**

	#301		#302		#303		#304	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Lily	0.94	0.82	0.89	0.65	0.65	0.71	0.95	0.97
RiMOM	0.76	0.69	0.72	0.65	0.76	0.88	0.90	0.97
ASMOV	0.89	0.77	0.61	0.46	0.73	0.83	0.90	0.92
EW+	0.80	0.72	0.60	0.54	0.43	0.85	0.77	0.85
CDW+	0.91	0.82	0.72	0.65	0.42	0.83	0.85	0.93
CONTX+	0.89	0.80	0.72	0.65	0.36	0.73	0.86	0.95

Comparing to these powerful matching systems, our simple matchers obtain amazing results, especially the overall recall rates close to that of the best of these three systems. However, the precision rates of our three matchers are significantly low in the case of #303, where the amount of correct correspondences is 48, and the size of smaller ontology is 96, so the theoretical precision rate of our self-tuning matcher then will be or less than 48/96=0.50.

5. Stable Match method for self-tuning aggregator

5.1. Overview

As defined above, the aggregator aims to combine the outcomes of the multiple matchers. The key of self-tuning aggregator is the weights of the matchers involved, which also associates with quality evaluation of the matchers. AgreementMaker proposes a unique measure, which is proportional to the similarity values of selected mappings, while detecting and penalizing those matchers which tend to assign high similarity values too generously.

We present here another method. In the one-to-one matching case, given two ontology O_1 and O_2, e_1 from O_1, and e_2 from O_2, we define a *stable match* as both the similarity value from e_1 to e_2 and that from e_2 to e_1 are the maximum. In Figure 1, there are two stable matches, *matching("Inbook", "Inbook")* and *matching("Incollection", "Collection")*. *matching("Inbook", "Book")* is not a stable match, although its value is larger than that of *matching("Incollection", "Collection")*.

Actually, stable match is an idea similar to well-known *stable marriage*, modeling the candidate correspondences as weighted bipartite graph as well. We evaluate a matcher by its ability of finding stable matches, which also represents the preference of the matcher. For example, if a matcher finds more stable matches than others, we think this matcher is more suitable for these ontologies, and assign it larger weight in turn. Unlike the strategies adopted by other systems like Falcon-AO, using design-time rules to link the ontology and a preferred matcher, we then provide a method to magnify a matcher automatically when we find it capable of finding more stable matches.

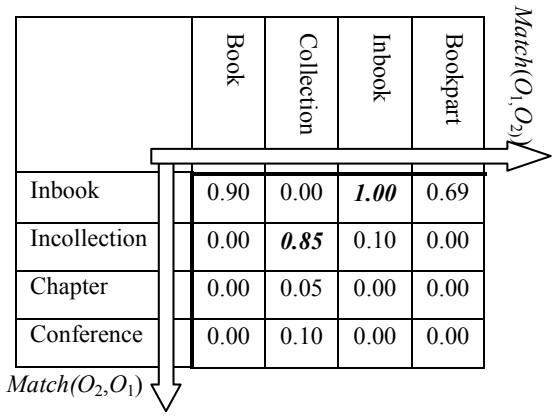

Figure 1. **Example of stable matching**

5.2. Self-tuning aggregator test

5.2.1. Test one. In the test of Section 4.2.1, the overall marks of three self-tuning matchers are CONTX+ the best, then EW+ and CDW+ in turn. If they are combined to get the final alignment, the aggregator should assign CONTX+ a highest weight, and then EW+ the second highest weight, and CDW+ the lowest weight.

We calculate the amount of stable matches found by the matchers, CONTX+ 30, EW+ 25, and CDW+ 20. The weights for these matchers are obtained then, CONTX+ 0.40, EW+ 0.33, and CDW+ 0.27.We find it is interesting that the better marking the matcher gets in this matching case, the more stable matches it finds out.

5.2.2. Test two. We run the aggregator in two rounds. In the first round, we turn off the MWBGM method, and in the second round, we turn it on. The results of two rounds are show in Figure 2 and Figure 3 as LiSTOMS-1, and LiSTOMS-2 separately.

The results are amazing. Although our system is a light-weighted self-tuning prototype with three simple matchers, the overall recall rates are leading, and the precision rates are competing. With MWBGM, we improve precision with only a little loss of recall in the #303 case.

6. Conclusions

With RMOMS, the Reference Model of Ontology Mapping System, we break down self-tuning ontology system problem into a few more feasible parts, and focus our research on two of them, the matchers and aggregator. We design LiSTOMS as our first prototype and evaluation platform of RMOMS, and extend *Maxium Weight Bipartite Graph Matching* method (presented in Section 4.1), as a generic self-tuning add-on for all matchers, and use *Stable Match* method for self-tuning aggregator. Equipped with these methods, our light-weighted system, LiSTOMS has already shown its leading recall rate and competing precision rate.

Table 3. **Results of precision**

	#301	#302	#303	#304
Lily	0.94	0.89	0.65	0.95
RiMOM	0.76	0.72	0.76	0.9
ASMOV	0.89	0.61	0.73	0.92
LiSTOMS-1	0.88	0.67	0.36	0.78
LiSTOMS-2	0.91	0.72	0.43	0.86

Table 4. **Results for Recall**

	#301	#302	#303	#304
Lily	0.82	0.65	0.71	0.97
RiMOM	0.69	0.65	0.88	0.97
ASMOV	0.77	0.46	0.83	0.97
LiSTOMS-1	0.82	0.65	0.88	0.96
LiSTOMS-2	0.82	0.65	0.85	0.95

7. References

[1] J. Euzenat and P. Shvaiko, *Ontology Matching*, Springer Berlin Heidelberg, 2007.

[2] P. Shvaiko and J. Euzenat, "Ten Challenges for Ontology Matching," *On the Move to Meaningful Internet Systems: OTM 2008*, Springer Berlin / Heidelberg, 2008.

[3] M. Sayyadian, Y. Lee, A. Doan, and A.S. Rosenthal, "Tuning schema matching software using synthetic scenarios," *Proceedings of the 31st International Conference on Very Large Data Bases*, Trondheim, Norway: VLDB Endowment, 2005, pp. 994-1005.

[4] Y. Lee, M. Sayyadian, A. Doan, and A.S. Rosenthal, "eTuner: tuning schema matching software using synthetic scenarios," *The VLDB Journal*, vol. 16, 2007, pp. 97-122.

[5] A. Doan, P. Domingos, and A.Y. Halevy, "Reconciling schemas of disparate data sources: a machine-learning approach," *SIGMOD Rec.*, vol. 30, 2001, pp. 509-520.

[6] M. Ehrig, S. Staab, and Y. Sure, "Bootstrapping ontology alignment methods with APFEL," *Special interest tracks and posters of the 14th international conference on World Wide Web*, Chiba, Japan: ACM, 2005, pp. 1148-1149.

[7] N. Jian, W. Hu, G. Cheng, and Y. Qu, "Falcon-AO: Aligning Ontologies with Falcon," *Proceedings of the K-Cap Workshop on Integrating Ontologies*, CEUR-WS.org, 2005, pp. 85-91.

[8] J. Tang, J. Li, B. Liang, X. Huang, Y. Li, and K. Wang, "Using Bayesian decision for ontology mapping," *Web Semantics: Science, Services and*

Agents on the World Wide Web, vol. 4, 2006, pp. 243-262.

[9] I. F.Cruz, F.P. Antonelli, and C. Stroe, "Efficient Selection of Mappings and Automatic Quality-driven Combination of Matching Methods," *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) Collocated with the 8th International Semantic Web Conference (ISWC-2009)*, CEUR-WS.org, 2009.

[10] A. Gal and P. Shvaiko, "Advances in Ontology Matching," *Advances in Web Semantics I*, 2009.

[11] Y.R. Jean-Mary, E.P. Shironoshita, and M.R. Kabuka, "Ontology matching with semantic verification," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, Sep. 2009, pp. 235-251.

[12] D. Aumueller, H.H. Do, S. Massmann, and E. Rahm, "Schema and ontology matching with COMA++," *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, ACM New York, NY, USA, 2005, pp. 906-908.

[13] A. Valarakos, V. Spiliopoulos, and G. Vouros, "AUTOMS-F: A Framework for the Synthesis of Ontology Mapping Methods," *Networked Knowledge - Networked Media*, Springer Berlin / Heidelberg, 2009, pp. 45-59.

[14] H. Do and E. Rahm, "COMA: a system for flexible combination of schema matching approaches," *Proceedings of the 28th international conference on Very Large Data Bases*, Hong Kong, China: VLDB Endowment, 2002, pp. 610-621.

[15] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," vol. 10, 1966, pp. 707-710.

[16] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, Las Cruces, New Mexico: Association for Computational Linguistics, 1994, pp. 133-138.

[17] G. Stoilos, G. Stamou, and S. Kollias, "A String Metric for Ontology Alignment," *The Semantic Web – ISWC 2005*, Springer Berlin / Heidelberg, 2005, pp. 624-637.

[18] Z. Zhen, J. Shen, and S. Lu, "WCONS: An ontology mapping approach based on word and context similarity," *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT Workshops 2008*, Sydney, NSW, Australia: Inst. of Elec. and Elec. Eng. Computer Society, 2008, pp. 334-338.

[19] V.V. Raghavan and S.K.M. Wong, "A critical analysis of vector space model for information retrieval," *Journal of the American Society for Information Science*, vol. 37, 1986, pp. 279-287.

[20] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching," *Proceedings of the 18th International Conference on Data Engineering*, IEEE Computer Society, 2002, p. 117.

[21] Y. Chen and F. Fonseca, "A Bipartite Graph Co-Clustering Approach for Ontology Mapping," *Proceedings of the Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data. Colocated with the Second International Semantic Web Conference (ISWC-03)*, CEUR-WS.org, 2003.

[22] H.W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics*, vol. 52, 2005, pp. 7-21.

[23] R.M. Karp, "An algorithm to solve the m × n assignment problem in expected time O(mn logn)," *Networks*, vol. 10, 1980, pp. 143-152.