Towards Combining Ontology Matchers via Anomaly Detection

Alexander C. Müller and Heiko Paulheim

University of Mannheim, Germany
Research Group Data and Web Science
heiko@informatik.uni-mannheim.de,alexanda@mail.uni-mannheim.de

Abstract. In ontology alignment, there is no single best performing matching algorithm for every matching problem. Thus, most modern matching systems combine several *base matchers* and aggregate their results into a final alignment. This combination is often based on simple voting or averaging, or uses existing matching problems for learning a combination policy in a *supervised* setting. In this paper, we present the *COMMAND* matching system, an *unsupervised* method for combining base matchers, which uses anomaly detection to produce an alignment from the results delivered by several base matchers. The basic idea of our approach is that in a large set of potential mapping candidates, the scarce actual mappings should be visible as anomalies against the majority of non-mappings. The approach is evaluated on different OAEI datasets and shows a competitive performance with state-of-the-art systems.

Keywords: Ontology Alignment, Anomaly Detection, Outlier Detection, Matcher Aggregation, Matcher Selection

1 Introduction

In ontology matching, there is only rarely a *one size fits all* solution. Ontology matching problems differ along many dimensions, so that a matching system that performs well on one dataset does not necessarily deliver good results on another one. To overcome this problem, many ontology matching tools combine the results of various base matchers, i.e., individual matching strategies. However, this approach gives way to a new problem, i.e., how to *combine* the results of the base matchers in a way that the combination suits the problem at hand [7]. Solutions proposed in the past range from simple voting to supervised learning.

In this paper, we propose to use *anomaly* or *outlier detection* for the problem of matcher combination. Anomaly detection is the task of finding those data points in a data set that deviate from the majority of the data [1]. The underlying assumption is that given a large set of mapping candidates (e.g., the cross product of ontology elements from the ontologies at hand), the *actual* mappings (which are just a few) should stand out in one way or the other. Thus, it should be possible to discover them using anomaly detection methods. We show that it is possible to build a competitive matching system combining the results of more than 25 base matchers using anomaly detection.

2 Approach

COMMAND is a novel approach for dynamically selecting and combining ontology matchers via anomaly detection. The overall architecture is depicted in Fig. 1. The platform was implemented in Scala, the code is available on github under an open-source license.[1]

2.1 Base Matching and Matcher Selection

First, all base matchers that are based on local information of each ontology entity are executed. The entities of the target and source ontology are matched in a pair-wise fashion. This step matches *Classes*, *DataProperties* and *ObjectProperties* pairwise and independently.

After this the first *feature vector* is analyzed and an uncorrelated feature subset is extracted. The results of those uncorrelated matchers are used as the input similarities for the *structural matchers*.

The result of the *structural matchers* is joined with the element level matcher result to create a *feature vector*. Since some of the features might be redundant or not vary in their values and thus do not contribute to the final matching, we remove results with little variation, correlated results, and also support PCA for computing meaningful linear combinations of base matcher results.

The current version of COMMAND implements a large variety of element and structure level techniques. Those encompass 16 string similarity metrics, five external metrics based on WordNet and corpus linguistics, and five structural matching techniques, such as similarity flooding.

2.2 Aggregation by Anomaly Detection

The next step is the aggregation of the base matcher results into a final matching score for all correspondences. We perform this step by detecting outlying datapoints in the *feature vector space*, and using this score as a measure of similarity. The anomaly analysis and score normalization are performed separately for classes, data properties, and object properties.

To compute outlier scores, we apply *anomaly analysis techniques* on the feature vector representations. In this paper, we use three different techniques: A *k-nearest-neighbor based method (KNN)* that computes the anomaly score of a data point based on the average euclidean distances[2] to its nearest neighbors, a cluster-based method that calculates the unweighted *cluster-based local anomaly factor (CBLOF)* based on a given clustering scheme produced by an arbitrary clustering algorithm [5], and the *Replicator Neural Networks (RNN)* method, which trains a neural network capturing the patterns in the data, and identifies those data points not adhering to those patterns [4].

[1] https://github.com/dwslab/COMMAND

[2] Note that since we expect all base matcher scores to fall in a $[0; 1]$ interval, using geometrical distance measures in that space is feasible.

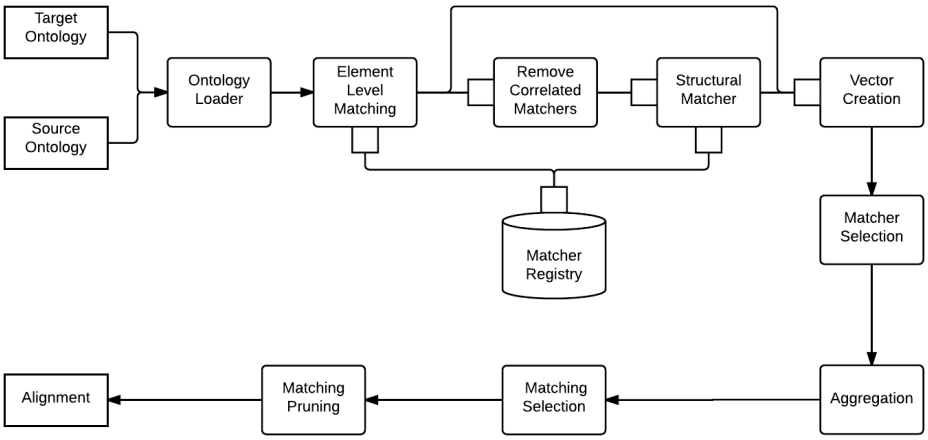

Fig. 1. Overview of the COMMAND pipeline

2.3 Matching Selection and Repair

The result of the previous step is a set of candidates, which does not necessarily form a semantically coherent mapping. After applying a threshold to the results of classes, data and object properties, the mapping may be refined by the *Hungarian method*, a *greedy selection*, or a *fuzzy greedy selection* [2]. Furthermore, logical consistency may be ensured by running the *ALCOMO* mapping post-processing system [6].

3 Evaluation

To evaluate the COMMAND approach, we use the *benchmark*, *conference*, and *anatomy* of the Ontology Alignment Evaluation Initiative (OAEI) 2014 [3].

We compare the results of COMMAND to three baselines. *Single best global* refers to the single base matcher that performs best on the given test case (i.e., conference, benchmark, and anatomy), using the optimal global threshold. *Majority vote* performs a voting across all base matchers, again using the best global threshold. *Single best local* selects the best base matcher for each problem.[3]

Furthermore, we compare COMMAND to the contestants of the OAEI 2014 initiative. To make that comparison fair, we use one global parameter set for each variant across all three OAEI datasets, instead of per dataset settings.

Tables 1, 2, and 3 depict the results of COMMAND on the OEAI datasets, once with and once without the use of ALCOMO. For anatomy, we restrict ourselves to the CBLOF variant and a subset of eight element-level matchers due to reasons of runtime. Except for the *Single best local* baseline (which is informative and not a baseline that can actually be implemented), COMMAND outperforms all baselines. When comparing COMMAND to the results of OAEI

[3] Note that in practice, it would not be possible to implement a matcher like *Single best local*. We only report it for informative purposes.

Table 1. Results on the OAEI biblio benchmark dataset. The table reports macro average recall, precision, and F-measure, with micro average values in parantheses.

Approach	without ALCOMO			with ALCOMO		
	Precision	Recall	F1	Precision	Recall	F1
Single best global	.754 (.733)	.557 (.521)	.641 (.609)	.779 (.761)	.548 (.521)	.644 (.619)
Majority vote	.510 (.472)	.570 (.544)	.538 (.505)	.524 (.487)	.463 (.443)	.491 (.464)
Single best local	.788 (.718)	.632 (.616)	.702 (.663)	.835 (.798)	.610 (.584)	.705 (.674)
CBLOF + PCA	.833 (.983)	.444 (.470)	.579 (.636)	.832 (.981)	.432 (.457)	.568 (.624)
CBLOF + RC	.844 (.982)	.466 (.461)	.600 (.627)	.844 (.982)	.457 (.449)	.593 (.617)
k-NN + PCA	.868 (.977)	.547 (.550)	**.672** (.704)	.871 (.975)	.480 (.459)	**.619 (.624)**
k-NN + RC	.847 (.967)	.549 (.556)	.666 (**.706**)	.835 (.984)	.463 (.442)	.596 (.610)
RNN + PCA	.881 (.991)	.466 (.443)	.610 (.612)	.859 (.965)	.324 (.253)	.470 (.401)
RNN + RC	.877 (.988)	.470 (.448)	.612 (.616)	.877 (.987)	.471 (.450)	.613 (.618)

Table 2. Results on the OAEI conference dataset. The table reports macro average recall, precision, and F-measure, with micro average values in parantheses.

Approach	without ALCOMO			with ALCOMO		
	Precision	Recall	F1	Precision	Recall	F1
Single best global	.641 (.784)	.591 (.611)	.615 (.687)	.640 (.783)	.591 (.611)	.615 (.686)
Majority vote	.874 (.949)	.537 (.552)	.665 (.698)	.874 (.949)	.537 (.552)	.665 (.698)
Single best local	.651 (.795)	.602 (.625)	.626 (.700)	.650 (.793)	.602 (.625)	.625 (.699)
CBLOF + PCA	.693 (.678)	.636 (.613)	.663 **(.644)**	.737 (.715)	.625 (.600)	**.676 (.652)**
CBLOF + RC	.702 (.693)	.607 (.577)	**.651** (.630)	.761 (.752)	.588 (.557)	.663 (.640)
k-NN + PCA	.718 (.712)	.572 (.534)	.636 (.610)	.797 (.782)	.557 (.518)	.656 (.623)
k-NN + RC	.710 (.702)	.574 (.541)	.635 (.611)	.781 (.769)	.530 (.492)	.631 (.600)
RNN + PCA	.829 (.815)	.528 (.492)	.645 (.613)	.748 (.699)	.617 (.587)	**.676** (.638)
RNN + RC	.820 (.805)	.527 (.489)	.641 (.608)	.819 (.804)	.524 (.485)	.639 (.605)

2014, we can find that the system, using CBLOF and PCA, and alignment repair with ALCOMO, would score on rank on a shared fifth rank (with XMap2) for the benchmark track, on rank four for the conference track (between LogMap-C and XMap), and on rank six (between LogMap-C and MaasMatch) for the anatomy track.

The runtime of *COMMAND* is assessed by measuring the time of a complete end-to-end pipeline execution. The general time complexity of *COMMAND* is quadratic to the size of the input ontologies. Additionally, the time consumption of the individual steps is measured. The results are depicted in table 4.

4 Conclusion and Outlook

In this paper, we have introduced a novel approach using anomaly detection for combining the results of different ontology matchers into a final aggregated matching score.

Overall, *COMMAND* performs an efficient *matcher selection* that only considers matchers that contribute to the final result, and uses *anomaly detection* as an unsupervised method for aggregating base matcher results. It is superior

Table 3. Results on the OAEI anatomy dataset.

Approach	without ALCOMO			with ALCOMO		
	Precision	Recall	F1	Precision	Recall	F1
Single best local/global	.920	.773	.840	.918	.740	**.820**
Majority vote	.932	.606	.735	.931	.597	.727
CBLOF + PCA	.892	.728	**.801**	.911	.741	.817
CBLOF + RC	.839	.664	.742	.832	.725	.775

Table 4. Average runtime in seconds of COMMAND

Dataset	Ø total	Ø t vector creation	Ø t aggregation	Ø t extraction
Conference	69.267	53.580	15.683	0.004
Benchmarks	52.880	44.026	8.850	0.004
Anatomy	18,746.510	11,595.601	5,922.478	1,228.431

to a simple majority vote baseline and performs in the range of state of the art matching tools. Furthermore, the possibility to use principal component analysis for feature space transformation also allows for implicitly computing relevant linear combinations of matcher scores.

The evaluation has been carried out on three OAEI datasets. For *conference* and *benchmarks*, the system achieved competitive performances in comparison to other OAEI participants. The results on the *anatomy* track showed that, since only a reduced configuration could be used with sub-optimal results, that more memory-efficient implementations are still required for fully exploiting the capabilities of COMMAND.

Furthermore future work will include the inclusion of other anomaly detection approaches, like angle-based methods, as well as other score normalization methods.

References

1. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Computing Surveys (CSUR) 41(3) (2009)
2. Do, H.H., Rahm, E.: Coma: A system for flexible combination of schema matching approaches. In: Proceedings of the 28th International Conference on Very Large Data Bases. pp. 610–621. VLDB '02, VLDB Endowment (2002)
3. Dragisic, Z.e.a.: Results of theontology alignment evaluation initiative 2014. In: International Workshop on Ontology Matching. pp. 61–104 (2014)
4. Hawkins, S., He, H., Williams, G., Baxter, R.: Outlier detection using replicator neural networks. In: Data warehousing and knowledge discovery, pp. 170–180. Springer (2002)
5. He, Z., Xu, X., Deng, S.: Discovering cluster-based local outliers. Pattern Recognition Letters 24(9), 1641–1650 (2003)
6. Meilicke, C.: Alignment incoherence in ontology matching. Ph.D. thesis (2011)
7. Shvaiko, P., Euzenat, J.: Ontology matching: State of the art and future challenges. Knowledge and Data Engineering, IEEE Transactions on 25(1), 158–176 (Jan 2013)