Mix'n'Match: Iteratively Combining Ontology Matchers in an Anytime Fashion

Extended report (accompanying OM2013 poster)

Simon Steyskal[1,2] and Axel Polleres[3,1]

[1] Siemens AG, Siemensstrasse 90, 1210 Vienna, Austria
[2] Vienna University of Technology, 1040 Vienna, Austria
[3] Vienna University of Economics & Business, 1020 Vienna, Austria

Abstract. We present a novel architecture for combining off-the-shelf ontology matchers based on iterative calls and exchanging information in the form of reference alignments. Unfortunately though, only a few of the matchers contesting in the past years' OAEI campaigns actually allow the provision of reference alignments in the standard OAEI alignment format to support such a combined matching process. We bypass this lacking functionality by
using simple URI replacement to "emulate" reference alignments in the aligned ontologies. While some matchers still consider classes and properties in ontologies aligned in such fashion as different, we experimentally prove that our iterative approach benefits from this emulation, achieving the best results in terms of F-measure on parts of the OAEI benchmark suite, compared to the single results of the competing matchers as well as their combined results. The new combined matcher – Mix'n'Match – integrates different matchers in a multi-threaded architecture and provides an anytime behavior in the sense that it can be stopped anytime with the best combined matchings found so far.

1 Introduction

Mapping between the internal models of software systems is a crucial task in almost all data and system integration scenarios, keeping computer scientists busy for the last decades with still no "silver bullet" in sight.

In quest for the said "silver bullet" [13] to solve such alignment problems, ontologies and ontology matching [2, 12, 19] seem to be a good starting point particularly for aligning object-oriented models, since ontologies can represent (and be extracted from) object-oriented models fairly naturally, where the hope is that the increasing number of off-the-shelf ontology matching tools can be used "out of the box" to propose reference alignments that can be tailored in a semi-automatic process.

In the past decade, the field of ontology matching has matured with many different ontology matchers having participated in the last years' OAEI [4] campaigns,

[4] http://oaei.ontologymatching.org/

exposing different strengths and weaknesses. The different tracks provided by the OAEI target different matching problems and since every matching tool has its specific special techniques and features, good performance in a specific sub-track often comes in hand with a bad performance in another one. So, intuitively, combining the results of a heterogeneous set of ontology matchers, taking the "best off-the-shelf matcher for the problem at hand" seems to be a promising approach for a "universal matcher" without the need for designing a new matcher from scratch.

Unfortunately, (i) neither deciding which matchers suit best for a set of given ontologies nor (ii) using ontology matchers in a semi-automatic human-guided matching process is trivial: for one, selecting the most suitable matcher(s) requires some preparatory work like interviews or tests [21] or background knowledge in terms of training sets for learning a classifier [9, 10].

On the other hand, for an interactive matching process, one needs to be able to guide a matcher by providing (or confirming) reference alignments, either provided by a domain expert directly or iteratively added by confirming matching results from an ontology matcher to be used as reference for supporting another call of the same or another ontology matcher.

In the present paper, we demonstrate the feasibility of a system that supports such a process in one go while still being unguided: we present a combined ontology matcher which, without the need to choose a single matcher nor requiring user interaction or training, performs better on average than a single matcher on a number of heterogeneous ontology matching problems (from the OAEI campaign). This is achieved by iteratively combining results of different matchers, and feeding them as support into subsequent runs of those matchers.

Our idea is strongly inspired by the definition of *ontology matching* presented in [11, 12]: that is, ontology matching can be summarized as a process which computes *alignments A'* for a pair of input ontologies *O1* and *O2*. As shown in Figure 1, there are three more components which can extend this basic definition of the matching process: (*i*) *input* or *reference alignments* which support the matching process with initial knowledge about the matching domain; (*ii*) resources either some sort of background knowledge, e.g. in terms of dictionaries, and (*iii*) parameters, e.g. for narrowing the results by using specific weights or thresholds.
In this paper we will particularly focus on the usage of *reference alignments* stemming from (combined) matching results from different matchers as well as *weighting parameters* (for deciding which combined alignments to approve) to support our iterative matching process. As for reference alignments, the existence of a standardized ontology alignment format[5] promoted by the Ontology Alignment Evaluation Initiative (OAEI) should enable the usage of off-the shelf ontology matchers in this process that support the said alignment format. Unfortunately though, in our experiments it turned out that only a few of the available matchers contesting in the past years' OAEI campaigns actually support the provision of

[5] `http://alignapi.gforge.inria.fr/format.html`

reference alignments, which is why we had to find a workaround to emulate such reference alignments.

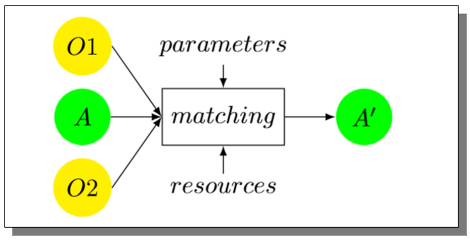

Fig. 1: Matching process from [12]

Structure of the Paper We start our paper with section 2 presenting a framework, which defines an architecture for a combined iterative matcher, that allows to plug existing matchers flexibly together and iteratively combines their results. Evaluation results on several benchmark problems from the OAEI campaign presented in Section 3 are very encouraging showing that our approach performs very stable, outperform each single matcher in many cases, or scoring comparably well to the best single matchers in other cases. We discuss ideas for future improvements in Section 4 before we conclude in Section 5.

Remark A preliminary version of this paper is to be published as a short paper in the 12th International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE 2013) [23]. New contributions in the present paper include: (i) a new streamlined architecture by using multi-threading for matcher execution instead of running ontology matchers sequentially in each iteration, (ii) better evaluation results, as well as (iii) implementing anytime behavior for time consuming matching tasks not yet tackled in [23].

2 The Mix'n'Match Framework

In this section, we present our combined approach to ontology matching, which we call "Mix'n'Match". Our goal is an approach that combines the above existing matchers with all their strengths and weaknesses in a unified manner. Instead of finding the one "best off-the-shelf matcher for the problem at hand" we aim at combining matcher results of different matchers. Intuitively, the idea of Mix'n'Match is very simple: starting from an empty set of alignments, we aim at iteratively supporting in each round, matchers with the combined results of other matchers found in previous rounds, aggregating the intermediate results of a heterogeneous set (portfolio) of ontology matchers.

Two main obstacles need to be overcome to make this approach feasible.

Result aggregation: In each round we need to decide which alignments to keep and which ones to ignore from the union of new alignments found by all considered matchers.

Reference alignments: among the state-of-the-art ontology matchers from previous OAEI contests that we tested, only one supported reference alignments as inputs, so we need to find another way to "inject" alignments in a non-obtrusive way[6] into existing matchers.

As for result aggregation, for the time being, we have chosen a straightforward strategy based on a simple majority vote, that is, all alignments confirmed by a majority of matchers above a certain threshold prevail in each matching round.

Adding the lacking support for reference alignments (without touching the matchers' source code) was not that easy. Our first idea was to add reference alignments explicitly in the form of OWL equivalences, using properties `owl:equivalentClass` (for matching concepts), `owl:equivalentProperty` (for matching properties), or `owl:sameAs` (for matching individuals).

Example 1. Assuming two ontologies from the education domain, with matched classes *Teacher* and *Professor* in a first matching round, implementing this strategy would mean to simply add to both ontologies the following OWL statement after this first matching round (and hoping it would be considered for subsequent reruns of the considered matchers):

```
<http://ontology.org/onto1/Teacher> owl:equivalentClass
<http://ontology.org/onto2/Professor> .
```

However, this approach to "emulate" reference alignments proved unsatisfactory, since we would be referring in $O1$ to entities from $O2$ and vice versa. First, such referring would need additional definitions (`owl:Class`, `owl:DatatypeProperty`, or `owl:ObjectProperty` of the referred entities in the respective other ontology of all aligned entities, since otherwise the such enriched ontology would no longer satisfy the syntactic restrictions of OWL DL: obviously, this is inconvenient, because it would lead to significant overhead essentially redefining already existing entities. Somewhat surprisingly, an alternative, much simpler approach to emulating reference alignments proved to be much more effective than adding OWL equivalence axioms: instead of "axiomatizing" alignments using OWL, we just created for each reference alignment a combined new URI for the matched entities, which was then replaced within both $O1$ and $O2$. Let us illustrate this replacement in our running example.

Example 2 (cont'd). Based on a possible found alignment in round 1 we modify both ontologies by replacing the URIs of the two classes *Teacher* and *Professor*, assuming an alignment between this two classes was found, by a unified one *<http://example.org/MixMatch/Teacher_Professor>*, cf. Listings 1+2 showing

[6] In order to keep our framework general and extensible, we do not want to modify the code of the ontology matchers or interfer in some other tool-specific way with the matchers used in our framework.

snippets from both ontologies after replacement; the expectation is that this replacement in a subsequent matching round helps to find additional alignments, such as in this case for instance the *name* and *hasName* properties, and so forth.

Listing 1: Ontology *Ont*1 after enrichment

```
...
@prefix mm: <http://example.org/MixMatch/> .
...
mm:Teacher_Professor a owl:Class.

ont1:name a owl:DatatypeProperty;
    rdfs:domain mm:Teacher_Professor; rdfs:range xsd:string
...
```

Listing 2: Ontology *Ont*2 after enrichment

```
...
@prefix mm: <http://example.org/MixMatch/> .
...
mm:Teacher_Professor a owl:Class.

ont2:hasName a owl:DatatypeProperty;
    rdfs:domain mm:Teacher_Professor; rdfs:range xsd:string
...
```

A schematic overview of the overall Mix'n'Match workflow is shown in Figure 2; we briefly explain each step in the following.

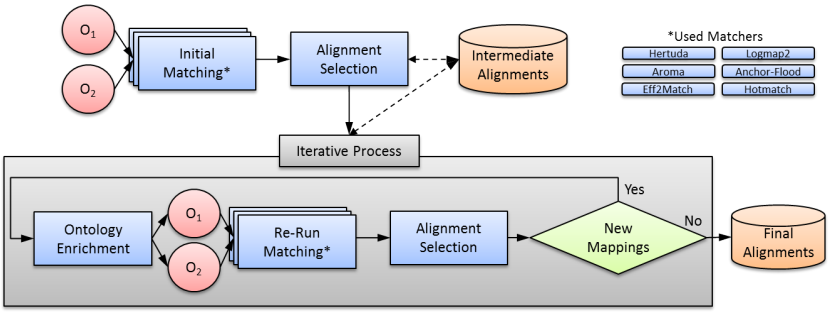

Fig. 2: Workflow of Mix'n'Match

Initial Matching In the initial matching step all participating ontology matchers try to find alignments of the two unaltered base ontologies.

Alignment Combination The combination of the alignments, especially the choice of those which are used for the enrichment step is – as mentioned above –

based on majority votes. By only accepting alignments which were found by a large number of heterogeneous matching tools we aim to ensure a high precision of the found alignments and therefore try to emulate reference alignments provided by a human domain expert. Although Mix'n'Match would support the definition of an alignment confidence threshold as additional parameter (i.e, only allowing alignments over a specific threshold to pass) we set this threshold per default to 0 in our experiments: since the calculation of confidence values is not standardized across matchers and some matchers only produce boolean confidence values (e.g. [15]). Other result aggregation methods may be conceivable here, like taking the individual performance of off-the-shelf matchers on specific matching tasks into account [4, 22], but since this approach would lead to a more inflexible alignment process this issue needs more detailed investigations in future versions of Mix'n'Match.

Enrichment Upon the approved aggregated alignments, enrichment of the ontologies (which imitates the "injection" of reference alignments) takes place, implementing the simple URI replacement sketched above: for every pair of matched entities in the set of aggregated alignments, a merged entity URI is created and will replace every occurrence of the matched entities in both ontologies (cf. Example 2). This approach is motivated by the assumption that if two entities were stated as equal by the majority of ontology matchers, their URI can be replaced by an unified URI, stating them as equal in the sense of URIs as global identifiers. Note that, despite the fact that most matchers seem to ignore URIs as unique identifiers of entities, our experiments showed that URI replacement was effective in boosting the confidence in such asserted alignments in almost all considered matchers.[7] Again, other enrichment techniques – as mentioned before, for instance, for matchers supporting OWL, reference alignments could be encoded by enrichment with respective OWL axioms – could be conceivable and would fit into the generic framework of Mix'n'Match. Note also that in the current version of Mix'n'Match only one-to-one and equality alignments were considered, we leave more complex alignments (such as 1:n mappings) for future work.

Iteration & Re-Run Matching After enrichment we start a new matching round, i.e., we restart the individual matchers on the enriched ontologies, gathering new reference alignments, followed by another alignment combination step, and so forth, until a fix point is reached, in the sense that the alignment combination step produces no further new alignments. We deploy a simple distribution strategy using multi-threading to run the matchers in each round in parallel, if possible (depending on available processors).

[7] Deliberately leaving out internal details of the respective matchers, we may only conjecture here that other features like string-matching were triggered after reference alignments being "asserted" by URI replacement, which consequently lead to further new alignments being found in subsequent calls.

Intermediate Results We collect the intermediate results of every finished off-the-shelf matcher in every iteration. Furthermore we keep track of every (non-contradictory) alignment found so far together with the number of individual matchers which have found this alignment in any previous matching round. This offers the possibility to interrupt the matching process at any time, retrieving only those alignments which have been found by the majority of the ontology matchers at the time the interruption has taken place. In contrast to other ontology matchers which offer this anytime behavior like MapPSO [1], we are not only restricted to gather alignment results of the last finished matching iteration, but use the alignment results of finished off-the-shelf matchers in the current matching round too.

Implementation We have implemented Mix'n'Match using Java for the general framework, and integrated existing matchers either by importing the respective Java sources, if available, or including .jar files (some of which were available through the SEAL[8] project).

3 Evaluation

We evaluated our framework using datasets provided by the OAEI campaigns, where Mix'n'Match proved able to outperform current ontology matching tools or at least competitive with the top-performing matchers (when only combining the matching results of the non-top-performers), thus encouraging further work on our approach.

As evaluation system we used an Intel Core Duo 3GHz with 4GB RAM and Windows 7 64 Bit as OS.[9]

3.1 Evaluation Datasets & Matcher Portfolio

As for evaluation, our intention was firstly to evaluate Mix'n'Match with the same benchmark datasets used as in [9], where the focus lies on evaluating a machine learning based approach for combining different ontology matchers; within [9] two benchmarks from different OAEI tracks are used, namely,

- parts of the OAEI benchmark track, where four real world ontologies (#301 to #304) are matched against one baseline ontology (#101).
- a part of the OAEI conference track, where various conference ontologies namely (*cmt, confOf, ekaw, edas, conference, iasted* and *sigkdd*) are all matched against each other.[10]

[8] http://www.seals-project.eu/

[9] We note that we expect even better results on a multi-core platform with more cores by the mentioned multi-threading.

[10] Note that in [9] only five of these ontologies were used, probably referring to an older instance of OAEI test data; we based our experiments on the OAEI 2012 datasets.

Secondly, in addition to these datasets we evaluated our approach on the OAEI anatomy track, primarily to check the runtime performance of Mix'n'Match on large datasets together with the applicability of anytime behavior within this use-case, where we expect an even longer runtime.

As for a representative portfolio of ontology matchers, we have considered a set of 6 ontology matchers mainly on the basis of their availability and ease of integration. That is, among the past OAEI participants, we have been looking for matchers (i) that showed good results in particular categories, (ii) capture a representative set of heterogeneous features and approaches and, last but not least, (iii) were available as Java source code or object code (.jar), which could be easily integrated in our framework; namely, we deployed *AROMA* [7], *Anchor-Flood* [15], *Eff2Match* [3], *LogMap2* [18], *HotMatch* [6] and *Hertuda* [16]. For matching large ontologies (>2000 entities), i.e., in the anatomy track, we excluded *HotMatch* and *Hertuda* from the Mix'n'Match portfolio since we otherwise ran into Java Heap Space / Memory Exceptions.

3.2 Evaluation Results

Benchmark				
Matcher	Precision	Recall	F-Measure	Time
Mix'n'Match	**93,98%**	**76,89%**	**84,58%**	**196486 ms**
Anchor-Flood	89,24%	78,51%	83,53%	1360 ms
Eff2Match	87,65%	75,47%	81,11%	34657 ms
Hertuda	75,13%	55,66%	63,94%	1077 ms
HotMatch	82,46%	45,54%	58,67%	4921 ms
Aroma†	59,06%	50,15%	54,24%	1723 ms
LogMap2	84,26%	31,44%	45,79%	5141 ms

Table 1: Aggregated results of the Benchmark Track ranked by F-Measure

Regarding the benchmark track 1, Mix'n'Match was able to outperform all tested and used ontology matchers, although it had the highest runtime which is not surprising given the iterative usage of every single off-the-shelf matcher. Unfortunately *AROMA* was not able to retrieve alignments for testcase 303 in our test setting (†in Table 1).

As stated in Table 2 on the conference track Mix'n'Match was able to retrieve the second highest F-Measure value of all used matchers. Nevertheless LogMap2 retrieved better results, but since our approach is primarily based on using majority vote for choosing alignments, single outstanding[11] ontology matchers don't significantly affect the results if we considering their alignments for Mix'n'Match.

[11] outstanding in terms of very good or very bad results

Conference				
Matcher	Precision	Recall	F-Measure	Time
LogMap2	78,81%	57,91%	66,76%	22072 ms
Mix'n'Match	**66,20%**	**59,91%**	**62,90%**	**648403 ms**
Hertuda	72,62%	51,01%	59,92%	1953 ms
HotMatch	69,66%	51,60%	59,29%	8994 ms
Anchor-Flood	45,16%	57,73%	50,68%	2451 ms
Eff2Match	34,43%	64,59%	44,91%	90649 ms
Aroma	30,85%	45,97%	36,92%	5667 ms

Table 2: Aggregated results of the conference track ranked by F-Measure

Anatomy				
Matcher	Precision	Recall	F-Measure	Time
Mix'n'Match	**94,29%**	**82,85%**	**88,2%**	1400142 ms
Logmap2	91,38%	84,63%	87,88%	16274 ms
Eff2Match	87,97%	82,98%	85,4%	85504 ms
Aroma	86,4%	68,73%	76,56%	18371 ms
AnchorFlood	87,93%	67,28%	76,23%	7073 ms

Table 3: Results of the anatomy track ranked by F-Measure

As Table 3 shows, Mix'n'Match again achieved the highest F-measure in the anatomy track. More remarkably, in Figure 3 we can observe the increasing F-Measure value of iterative matching in this track. This promising results show, that (i) our approach finds additional alignments in each matching round; (ii) was able to outperform the individual matchers used for the matching process and (iii) since Mix'n'Match has usually a quite bad runtime performance (1400142 ms for matching the anatomy track) but supports anytime behavior, it would have also been possible to abort the matching process at any intermediate time receiving already found alignments. Since the increase of the F-Measure value slows down in later matching rounds we would have been able to receive a value between 83,87%-85,95% if we would have aborted the matching process after the half of the actually needed matching time.

Remark We recognized slight differences between the evaluation values reported by OAEI and those measured by us for the tested matchers. Nonetheless, since our approach relies on the results of these other matchers, the relative improvement of the values should stay the same.

4 Related Work

The approach of using an iterative process which reruns matching techniques until no further alignment pairs were found is not new per se. Tools like Anchor-Flood [15] or ASMOV [17] are based on similar frameworks but in contrast to our

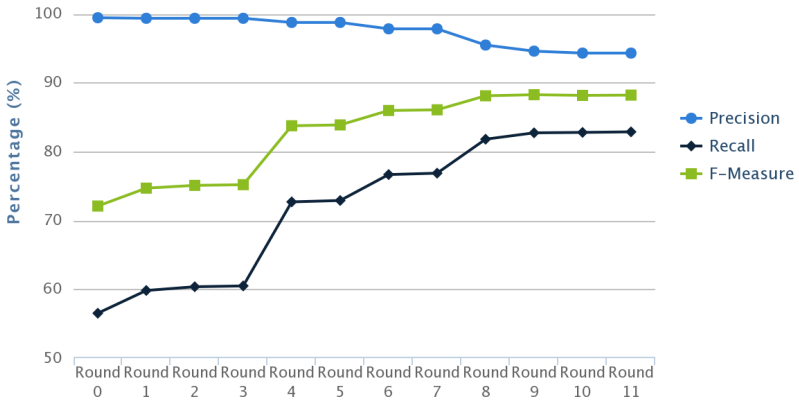

Fig. 3: Increasing F-Measure of Mix'n'Match in each iteration for the anatomy track

approach neither of them use off-the-shelf matchers for retrieving and combining alignments, but they combine several matching techniques in *one* tool. The benefit of combining whole matchers out of the box as in Mix'n'Match, instead of specifically implemented matching techniques is obvious: on the one hand we can highly increase the flexibility of Mix'n'Match since new single off-the-shelf matcher can easily be integrated into or segregated from the matching process and on the other hand since matching tools evolve over time and perform better and better by themselves, Mix'n'Match should also benefit from this evolution based on its framework.

Instead of combining ontology matchers in iterative matching steps, some approaches only focus on the combination of alignment sets of in a one-shot process. However, as such approaches do not rerun the matching process with the gathered additional knowledge again, we believe that they do not fully exploit the combined potential, which is somewhat confirmed by our evaluation results that needed several iterations to obtain a fixpoint. We note though that even one-shot combinations receive very good results especially when previously trained using machine learning techniques for the combination of the alignments [8,9,20]. While we focused on unguided combination so far, a combination of machine-learning techniques for results aggregation and our iterative approach is on our agenda.

Furthermore there exist some other approaches than a majority voite for combining the results of different ontology matching techniques which could be fairly easily adopted to work with our approach like in [14] where the authors propose an automated approach for weighting individual ontology matchers based on their importance on different matching tasks or in [4] were an automatic alignment evaluation by using quality measures is described.

In [5] the authors provide an approach for automatically configuring the parameters of off-the-shelf matchers and therefore optimize their performance; together with the approach proposed in [22] where unsupervised learning tech-

niques are used to find similarity parameters, these approaches could be used to improve the performance and flexibility of Mix'n'Match.

5 Conclusions

In this paper we have presented an alternative approach for combining off-the-shelf ontology matchers, using their combined alignment results for an iterative matching process. We have developed an architecture implementing the approach that parallelized the iterative runs of single matchers by using multi-threading and stores found reference alignments in a manner that allows to stop the iterative matching process at any time with the best alignments approved by a majority of matchers from a configurable portfolio of off-the-shelf matchers. Despite the fact, that such an approach would need the support of reference alignments only sparsely supported by current matchers, we have shown that it is possible to some extent to simulate such reference alignments by enrichment or simple URI replacement. Although we discovered that not all ontology matchers consider entities with the exact same URI as 100% identical, probably since string similarity measures seem to be part of nearly every investigated ontology matcher, URI replacement helped to exceed some internal confidence plateaus of found alignments and therefore accept them as correct alignments in our framework. Overall, the results we could obtain with the Mix'n'Match framework, which we have implemented upon these insights and verified experimentally, are encouraging and leave various promising roads for investigations in future work.

References

1. J. Bock, J. Hettenhausen. Discrete particle swarm optimisation for ontology alignment. *Information Sciences*, 192:152–173, 2012.
2. N. Choi, I.-Y. Song, H. Han. A survey on ontology mapping. *SIGMOD Record*, 35(3):34–41, 2006.
3. W. W. Khong Chua, J.-J. Kim. Eff2match results for oaei 2010. In *OM*, volume 689 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.
4. I.F. Cruz, F. Palandri Antonelli, C. Stroe. Efficient selection of mappings and automatic quality-driven combination of matching methods. In *ISWC International Workshop on Ontology Matching (OM) CEUR Workshop Proceedings*, volume 551, pages 49–60. Citeseer, 2009.
5. I.F. Cruz, A. Fabiani, F. Caimi, C. Stroe, M. Palmonari. Automatic configuration selection using ontology matching task profiling. In *The Semantic Web: Research and Applications*, pages 179–194. Springer, 2012.
6. T.T. Dang et al. Hotmatch results for OAEI 2012. In *Seventh International Workshop on Ontology Matching (OM 2012)*, 2012.
7. J. David, F. Guillet, H. Briand. Matching directories and OWL ontologies with aroma. In *15th ACM Int'l Conf. on Information and knowledge management*, CIKM '06, pages 830–831, New York, NY, USA, 2006. ACM.
8. A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, A.Y. Halevy. Learning to match ontologies on the semantic web. *VLDB J.*, 12(4):303–319, 2003.

9. K. Eckert, C. Meilicke, H. Stuckenschmidt. Improving ontology matching using meta-level learning. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009 Heraklion, pages 158–172, Berlin, Heidelberg, 2009. Springer-Verlag.
10. M. Ehrig, Y. Sure, S. Steffen. Bootstrapping ontology alignment methods with apfel. In *Proceedings of the 4th International Semantic Web Conference (ISWC)*. Springer, November 6-10 2005.
11. J. Euzenat, C. Meilicke, H. Stuckenschmidt, P. Shvaiko, C. Trojahn dos Santos. Ontology alignment evaluation initiative: Six years of experience. In *Journal of Data Semantics XV*, volume 6720 of *Lecture Notes in Computer Science*, pages 158–192. Springer, Berlin, Heidelberg, 2011.
12. J. Euzenat, P. Shvaiko. *Ontology Matching*. Springer, 2007.
13. D. Fensel. *Ontologies:: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2001.
14. M. Gulić, I. Magdalenić, B. Vrdoljak. Automated weighted aggregation in an ontology matching system. *International Journal of Metadata, Semantics and Ontologies*, 7(1):55–64, 2012.
15. M. Seddiqui Hanif, M. Aono. An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size. *J. Web Sem.*, 7(4):344–356, 2009.
16. S. Hertling. Hertuda results for oeai 2012. In *Seventh International Workshop on Ontology Matching (OM 2012)*, 2012.
17. Y.R. Jean-Mary, E.P. Shironoshita, M.R. Kabuka. Ontology Matching with Semantic Verification. *Web Semantics*, 7(3):235–251, September 2009.
18. E. Jiménez-Ruiz, B. Cuenca Grau, Y. Zhou. Logmap 2.0: towards logic-based, scalable and interactive ontology matching. In *Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences*, SWAT4LS '11, pages 45–46, New York, NY, USA, 2012. ACM.
19. Y. Kalfoglou, M. Schorlemmer. Ontology mapping: the state of the art. *Knowledge Eng. Review*, 18(1):1–31, 2003.
20. P. Maio, N. Bettencourt, N. Silva, J. Rocha. Evaluating a confidence value for ontology alignment. In *Proceedings of the 2nd International Workshop on Ontology Matching (OM-2007) Busan, Korea, November 11, 2007*, volume 304 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
21. M. Mochol, A. Jentzsch. Towards a rule-based matcher selection. In *Proceedings of the 16th international conference on Knowledge Engineering: Practice and Patterns*, volume 5268 of *Lecture Notes in Computer Science*, pages 109–119. Springer, 2008.
22. A. Nikolov, M. d'Aquin, E. Motta. Unsupervised learning of link discovery configuration. In *The Semantic Web: Research and Applications*, pages 119–133. Springer, 2012.
23. S. Steyskal and A. Polleres. Mix'n'match: An alternative approach for combining ontology matchers. In *Proceedings of the 12th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2013)*, Graz, Austria, September 2013. Short paper.

6 APPENDIX - Round Results

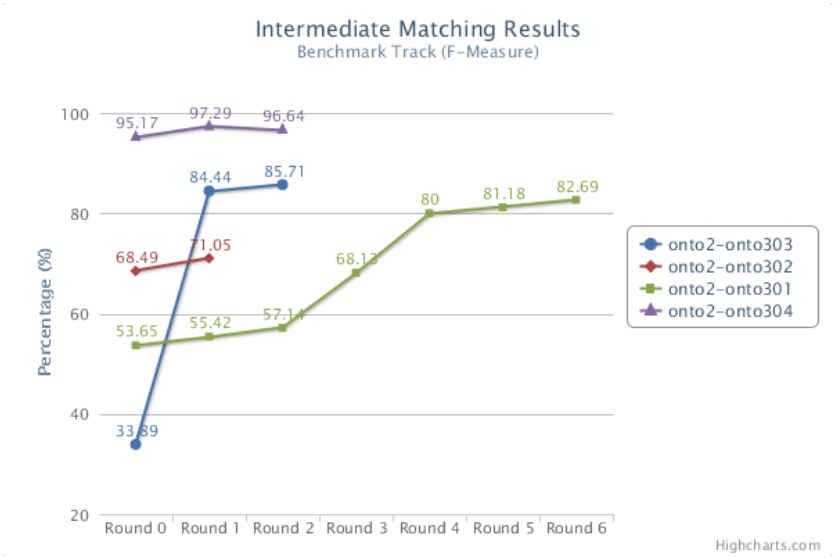

Fig. 4: Progress of F-measure value, matching the base ontology against all other ontologies

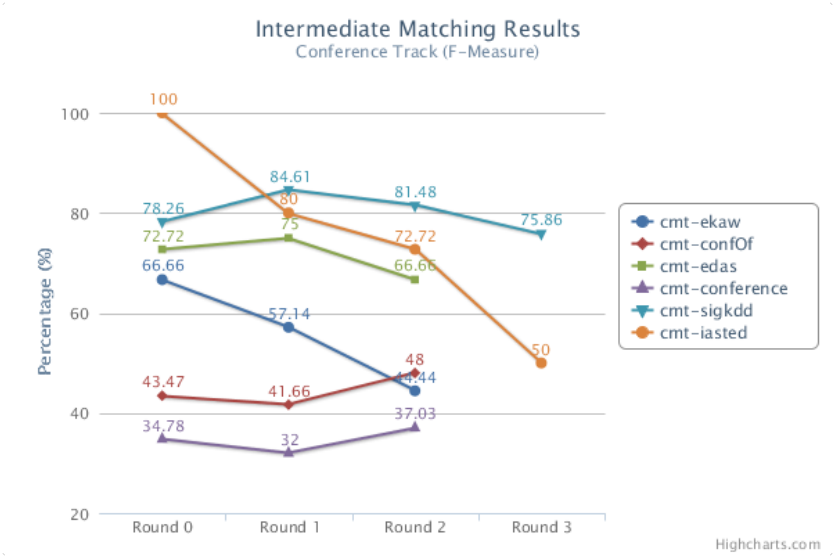

Fig. 5: Progress of F-measure value, matching cmt.owl against all other ontologies

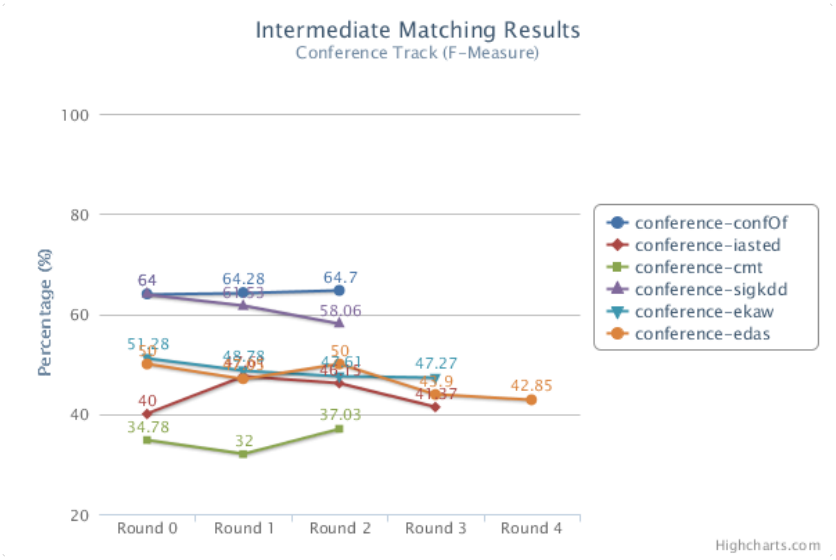

Fig. 6: Progress of F-measure value, matching conference.owl against all other ontologies

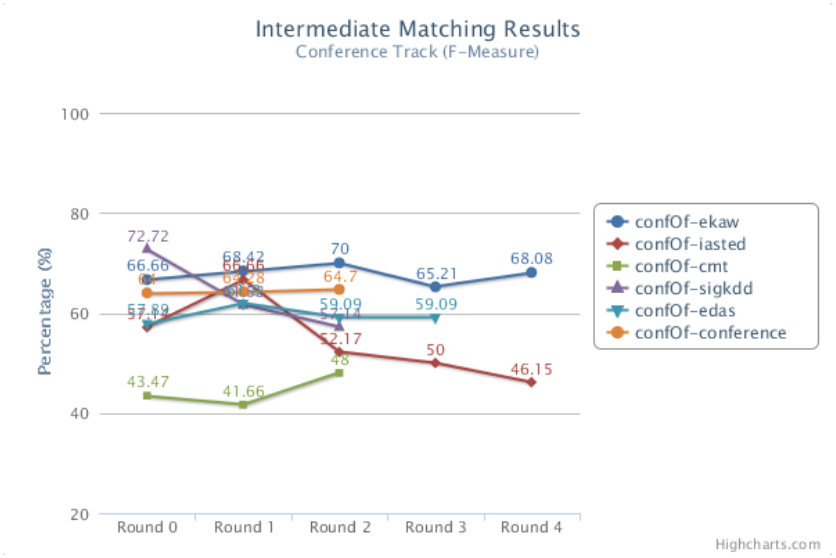

Fig. 7: Progress of F-measure value, matching confOf.owl against all other ontologies

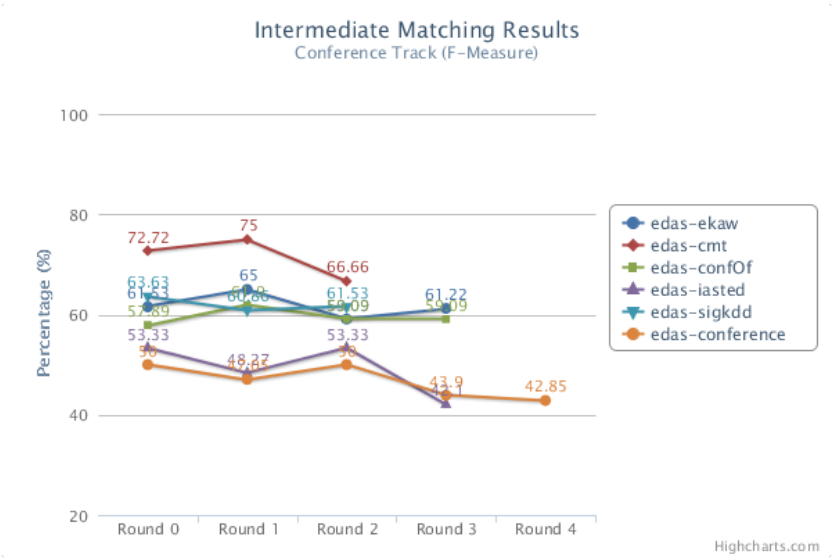

Fig. 8: Progress of F-measure value, matching edas.owl against all other ontologies

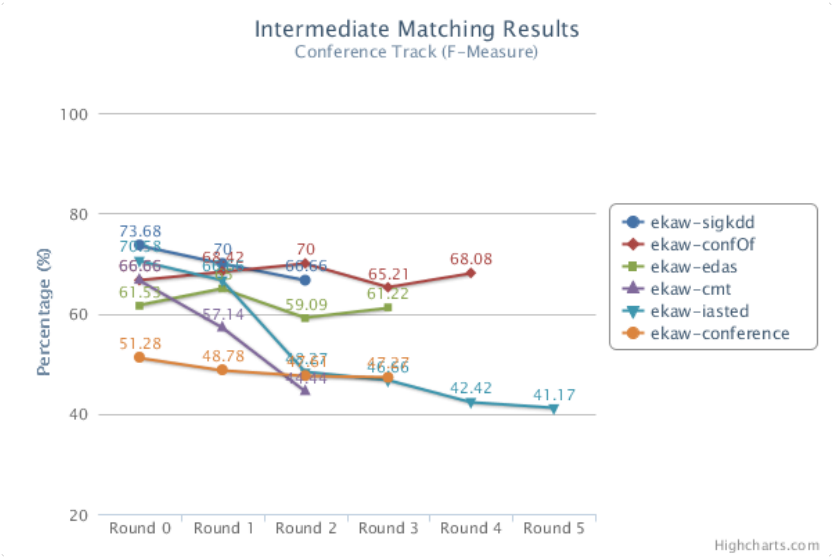

Fig. 9: Progress of F-measure value, matching ekaw.owl against all other ontologies

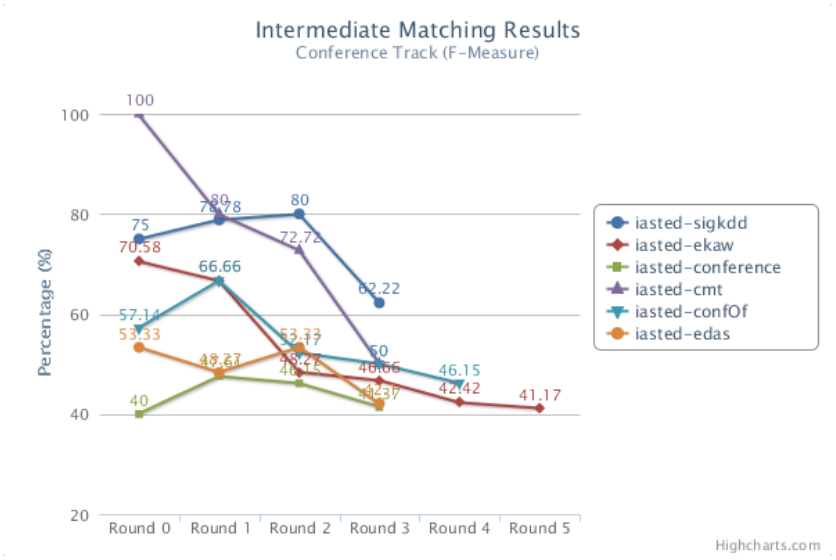

Fig. 10: Progress of F-measure value, matching iasted.owl against all other ontologies

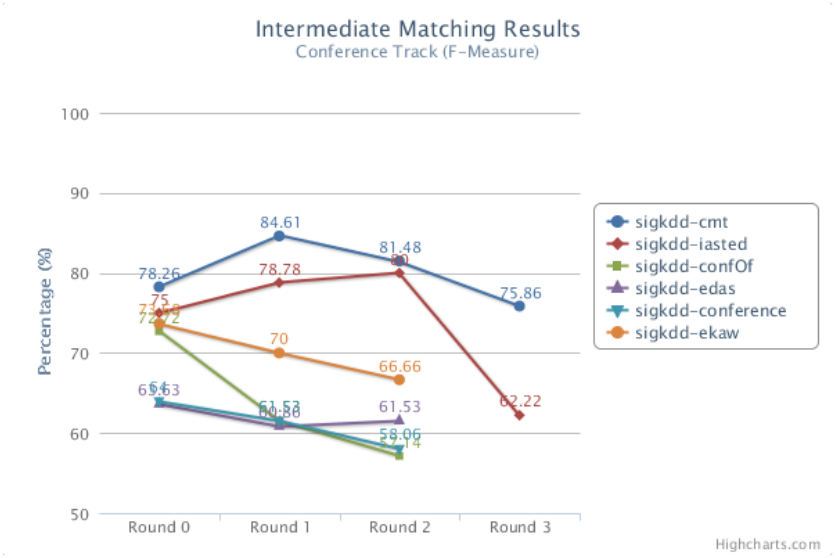

Fig. 11: Progress of F-measure value, matching sigkdd.owl against all other ontologies