HotMatch Results for OEAI 2012

Thanh Tung Dang, Alexander Gabriel, Sven Hertling, Philipp Roskosch,
Marcel Wlotzka, Jan Ruben Zilke, Frederik Janssen, and Heiko Paulheim

Technische Universität Darmstadt
{janssen,paulheim}@ke.tu-darmstadt.de

Abstract. HotMatch is a multi-strategy matcher developed by a group of students at Technische Universität Darmstadt in the course of a hands-on training. It implements various matching strategies. The tool version submitted to OAEI 2012 combines different basic matching strategies, both element-based and structure-based, and a set of filters for removing faulty mappings.

1 Presentation of the system

1.1 State, purpose, general statement

HotMatch[1] has been developed by a group of students in the course of a semantic web hands-on training conducted at TU Darmstadt. The students were asked to develop and implement different matching algorithms. For OAEI 2012, we have combined a large number of those matching algorithms into one tool. To give an overview of our approaches, all matchers are depicted in figure 1. In contrast to *matchers*, *filters* are used to remove mapping elements found by previous matchers.

1.2 Specific techniques used

HotMatch provides a library of different matching algorithms and filters.

Matching Algorithms
ElementStringMatcher is a simple string-based, element-level matcher on the element level. All labels, URI fragments and comments are extracted and tokenized. As a second step some stopwords are removed. To get a similary measure of two concepts, a cross product of labels, fragments and comments is calculated with the Damerau–Levenshtein distance.

GraphbasedUseClassMatcher is a graph based matcher. It operates on the structural level and needs some input alignment to have an initial mapping between classes.

Figure 2 gives an example of the mapping candidates. The properties X and Y are matched if the domain and range are equals respectively are aligned with a previous matcher. The confidence of the new mapping between the two Properties is the mean value between the confidence of mapping A to C and B to D.

[1] For **H**ands-**o**n **t**raining matcher

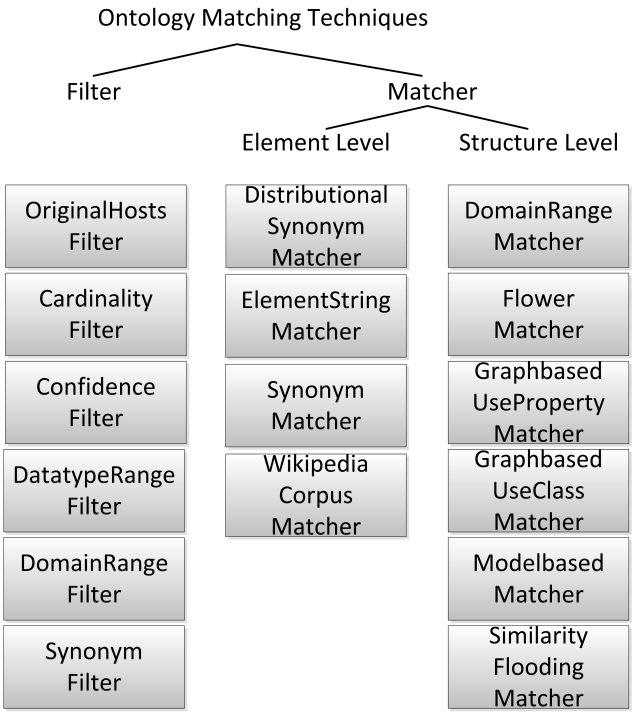

Fig. 1. Overview on the matching and filtering algorithms implemented in HotMatch.

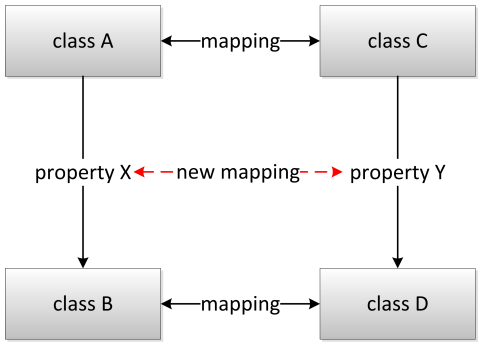

Fig. 2. New mapping of *GraphbasedUseClassMatcher*. Class A and C as well as B and D are already matched. Property X and Y is therefore also matched.

GraphbasedUsePropertyMatcher is a modification of *GraphbasedUseClassMatcher*. It uses properties from previous alignments instead of classes. If a property is matched from previous approaches, then the domain and range are also matched in a new alignment, inheriting the confidence mapping between the properties.

SimilarityFlooding implements the structural similarity flooding matching algorithm described in [3].

FlowerMatcher is a matching algorithm which combines a structural and an element-based approach. For each ontology class, its neighborhood (super and subclasses, properties that this class is a domain or range of) are regarded. From the names and labels of all the concepts in the neighborhood, a joint set of trigrams is computed. These sets are compared for determining the class similarity.

ModelbasedMatcher checks currently only if the union of the two ontologies plus the input mappings is valid. The implementation uses the *pellet* reasoner. In the future, this matcher is supposed to add extra mappings derived by reasoning, as well discard mappings that generate a contradiction.

DistributionSynonymMatcher and **WikipediaCorpusMatcher** are matchers using external resources, i.e., the online API *lanes*[2]. The distribution synonym matcher tries to identify synonyms based on distributional similarity, i.e., the similarity of the context in which two words occur [1]. The Wikipedia corpus matcher computes the percentage of Wikipedia pages on which two terms co-occur (similar to the approach discussed in [2]).

SynonymMatcher uses the online thesaurus *Big Huge Thesaurus*[3] to find mappings between concepts.

Filters

OriginalHostsFilter extracts the major host component of the input ontologies' URIs. If an alignment has other URI hosts than the major one, this alignment is removed. The remaining mappings are not changed. This filter is necessary, because an alignment like

$$\begin{gathered} < http://purl.org/dc/elements/1.1/description, \\ http://purl.org/dc/elements/1.1/description, \\ =, \\ 1.0 > \end{gathered}$$

is definitely true, but not contained in the reference alignments. In OAEI tracks, it will thus generate a false positive and reduce the mathcher's precision.

CardinalityFilter is a filter to enforce a $1:1$ alignment. If a resource from ontology one are matched to multiple resources from ontology two, then only the alignment with the highest confidence is selected. All other mappings are discarded. The same procedure is also applied for ontology two. The result of this filter is an alignment that relates each element from one ontology to at most one element from another ontology.

ConfidenceFilter is a simple filter that removes all alignments that have a smaller confidence than a given threshold.

[2] Language Analysis Essentials, `http://research.wilsonwong.me/lanes.html`

[3] `http://words.bighugelabs.com/`

DomainRangeFilter discards all alignments with non-matched domain and range. This is particularly useful for discarding inverses (e.g., *isReviewerOf* vs. *hasReviewer*), which receive high similarity scores with simple element-based techniques.

DatatypeRangeFilter checks only datatype properties. Matched properties hat have a different datatype (e.g., string vs. date) are discarded.

SynonymFilter has been implemented as a variant of the **SynonymMatcher** (see above). Since the latter has shown to produce a too large number of false positives (but with reasonable recall), it can also be used as a filter, e.g., on structural approaches for improving precision.

1.3 Adaptations made for the evaluation

The final matcher composition of the version submitted to OAEI 2012 is shown in figure 3. The threshold for confidence filter is set to $t = 0.7$. Note that not all matchers and filters discussed above are included in the final composition. We discarded all components that did not improve the system's accuracy and favored faster components over slower ones in case of ties.

All matchers are composed sequentially. The upper lane shows all matchers which generate new alignments. The lower one depicts all filters used to remove alignments that are not in the reference alignment to improve the precision value.

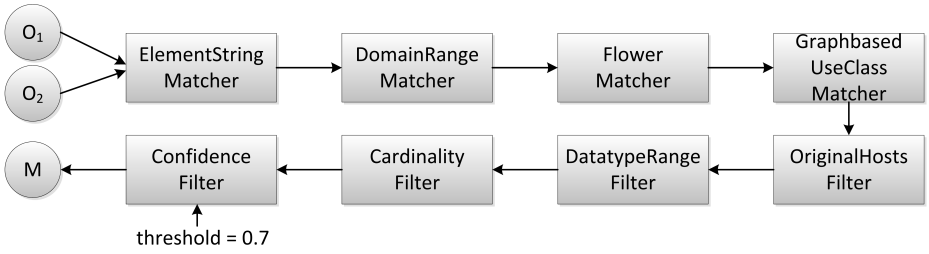

Fig. 3. Final composition for the evaluation

Although the filters only remove elements from the mapping generated by the matchers, they cannot be arbitrarily permuted. For example, the cardinality filter enforcing a 1:1 mapping will select the candidate with the highest threshold. If a mapping element with a higher threshold is filtered, e.g., by the OriginalHostsFilter, the selection will be different. Consider the following constellation for a mapping between ontology A and B, where B imports the FOAF ontology[4]:

$$< A\#person, B\#author, =, 0.7 > \quad (1)$$

$$< A\#person, foaf\#person, =, 0.8 > \quad (2)$$

[4] `http://xmlns.com/foaf/spec/`

Using the CardinalityFilter first would discard the first element, and the second one would be discarded by the OriginalHostsFilter. On the other hand, using the OriginalHostsFilter first would discard the second element, with the first one passing the CardinalityFilter.

1.4 Link to the system and parameters file

The tool version submitted to OAEI 2012 can be downloaded from `http://www.ke.tu-darmstadt.de/resources/ontology-matching/hotmatch`.

2 Results

2.1 Benchmark

HotMatch relies on string similarity to a large extent; although some structural measures are used later in the pipeline. Thus, it only performs well on those benchmark cases where names and labels are preserved. In particular, they show that the filters work quite effectively, since the precision only rarely drops below 0.95.

2.2 Anatomy

On the anatomy track, the performance of HotMatch is more or less the same as the string equivalence baseline[5]. In other words, the structure-based approaches do not improve the results much. This is not surprising as the structure-based approaches in HotMatch largely rely on `domain` and `range` definitions, which are not present in the Anatomy track. The reported runtime of 672 seconds shows an average behavior.

2.3 Conference

This track gives some insights into the strengths and weaknesses of *HotMatch*. In contrast to the anatomy track, the structure-based measures in HotMatch are capable of exploiting the `domain` and `range` definitions in the conference ontologies. For example, the structure-based algorithms provide some useful mappings, such as `hasAuthor = isWrittenBy` or `hasBeenAssigned = isReviewing`, but are also prone to produce false positives such as `Reviewer = MemberPC`, since both share a common super class. In terms of F-Measure, the results are comparable to *Baseline2*[6] (i.e., string matching with some pre-processing), but with a tendency to prefer recall over precision in comparison to that baseline, as the examples above show.

[5] `http://oaei.ontologymatching.org/2011.5/results/anatomy/index.html`

[6] `http://oaei.ontologymatching.org/2011.5/results/conference/index.html`

2.4 Multifarm

This matcher is not designed to work with multilingual ontologies. The results are accordingly low. Only some labels are equals in their translation like *person* in German as well as in English. Such resources are matched through string equality. Despite those occasional mappings, there is no correlation of the result quality and involved the languages' similarity – strangely enough, the best results are achieved for German-Chinese, two languages that are not known to be particularly similar.

2.5 Library

The mapping quality achieved by HotMatch on the library track is not as positive as on the other tracks. Possible reasons may be the absence of `domain` and `range` definitions (in fact, of properties in general), as for anatomy, and the presence of multi-lingual labels. As HotMatch does not respect languages, this may lead to false positives.

2.6 Large Biomedical Ontologies

HotMatch has been reported to have some problems of finishing the larger datasets in this track on time. As the matching process itself is rather light-weight, this may hint at efficiency issues of the implementation of HotMatch.

3 General comments

3.1 Comments on the results

The results show that with a multi-strategy approach using different simple matching strategies, reasonable results can be produced. There is a gap to more sophisticated systems – which is expected – but the results on the conference track also show that some of the more complex systems can be beaten.

3.2 Discussions on the way to improve the proposed system

One key feature of HotMatch is the ability to combine multiple matchers and filters. The final configuration submitted to OAEI has been found using extensive manual testing, however, it is a compromise which is supposed to produce reasonable results on most of the tracks.

Being able to individually assembling a configuration for each pair of ontologies would be an interesting option, thus, the system would clearly benefit from leveraging work in these fields [4, 5].

3.3 Comments on the OAEI 2012 Measures

In the current OAEI test cases, mapping elements that are correct but refer to concepts of other ontologies (like the example in Sect. 1.2) cause false positives, since they are not part of the reference alignment. In the HotMatch version for OAEI, we filter them manually, however, a real-world ontology matching system that returns those elements as well could equally make sense.

To circumvent this problem, the organizers might consider filtering mapping elements refering to concepts from other ontologies before computing precision.

4 Conclusion

In this paper, we have discussed the results for the HotMatch system, a multi-strategy matching system developed by students at Technische Universität Darmstadt in the course of a hands-on training. We have shown that the system provides reasonable results on most of the OAEI tracks and can compete with many state-of-the-art matching tools.

References

1. Harris, Z.S.: MathematicalStructuresOfLanguage. Wiley (1968)
2. Hertling, S., Paulheim, H.: Wikimatch - using wikipedia for ontology matching. In: Seventh International Workshop on Ontology Matching (OM 2012). (2012)
3. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: 18th International Conference on Data Engineering, IEEE (2002) 117–128
4. Mochol, M., Jentzsch, A., Euzenat, J.: Applying an analytic method for matching approach selection. In: Proceedings of the 1st International Workshop on Ontology Matching (OM-2006). (2006)
5. Ritze, D., Paulheim, H.: Towards an automatic parameterization of ontology matching tools based on example mappings. In: Sixth International Workshop on Ontology Matching. (2011)