MapSSS Results for OAEI 2011

Michelle Cheatham

[1] Wright State University, Dayton, OH, USA
michelle.cheatham@gmail.com

Abstract. MapSSS is very preliminary work on the feasibility of matching OWL -based ontologies in a manner that involves using only limited reasoning over the ontologies. This is the first year MapSSS has been a part of the OAEI competition, and it is hoped that these results will serve as a baseline for comparison with a more mature version of the algorithm a year from now.

1 Presentation of the system

MapSSS is an OWL ontology alignment algorithm designed to explore what can be accomplished using very simple similarity metrics rather than (or at least before) resorting to complex reasoning algorithms. OWL ontologies are treated as simple directed graphs with edges representing OWL relations and nodes representing classes, properties, and individuals. The basic algorithm consists of a syntactic, structural, and semantic metric. The metrics are conservative in the sense that a node that may match more than one node is ignored rather than risking making an error. These metrics are applied one after the other, and a positive result from any one of them is treated as a match. When a match is found, MapSSS attempts to capitalize on this by immediately recursing to look for matches among the immediate neighbors of the newly matched nodes.

1.1 State, purpose, general statement

MapSSS is meant to provided automated alignments of OWL-based ontologies. The basic algorithm is only two-thirds complete but still produces surprisingly good results on several OAEI test sets. All mappings found by MapSSS are currently considered to have a confidence level of 1.0. It would be relatively straightforward to adapt the metrics to use thresholds instead of requiring exact matches. MapSSS currently only supports finding equivalence relations – finding subsumption and other types of relations is a goal of future work.

1.2 Specific techniques used

The three 'S'es in MapSSS correspond to the three types of metrics the algorithm will eventually use: syntactic, structural, and semantic. The syntactic metric is a simple lexical comparison. The structural metric is a graph-based metric based on the direct neighbors of an entity and the edges that connect the entity to those neighbors. The semantic metric has not yet been implemented. We plan to use a Google Research account (http://research.google.com/university/search/) to determine whether or not a pair of entity labels are synonyms.

The main processing loop of MapSSS compares each entity in the first ontology to each entity in the second, first based on the syntactic metric, then the semantic, and finally the semantic metric. Whenever a match is found, the algorithm pauses its main loop execution and instead recurses on the newly matched nodes. At this point the metric treats the direct neighborhoods of the newly matched nodes as if they were the entire ontology. This improves recall because while there may be several matches for a given node within the entire opposing ontology, the one that is closest to an already-matched neighbor is most likely to be correct. The main loop repeats until no new mappings are added by any of the metrics.

Syntactic Metric

Levenstein distance is generally used for the syntactic mapping, though for the OAEI contest only exact matches are considered valid (i.e. the threshold is 1.0). Some minor pre-processing is done on the entity labels prior to running this metric. All labels are converted to lower case and camel case and underscores are converted to spaced words. For instance “oneTwo” and “one_two” are both converted to “one two”.

Structural Metric

As mentioned previously, MapSSS treats OWL ontologies as simple graphs. The structural metric acts on the direct neighborhood of the nodes in a candidate match. The metric only adds the candidate nodes to the mapping if they are the ONLY possible matches within the graph of subgraph being considered. The metric has the following constraints:

- The entities must be the same type to be considered possible matches (i.e. classes are only matched with classes, properties with properties, etc).

- Edge labels (e.g. subclass, domain, range, instance, allValuesFrom, someValuesFrom, minCardinality, maxCardinality, cardinality) must match exactly for the entities to be considered possible matches, and the corresponding neighbors at the end of those edges must be of the same type.

- If an entity in one ontology has a neighbor that is already part of the mapping, then the node that neighbor is mapped to must be a neighbor of any prospective match for this entity. This is similar to how the VF2 graph matching algorithm works. This constraint helps to ensure that the generated mapping is internally consistent and coherent.

Semantic Metric

We plan to use the Google Research API to query Google based on the potentially matching entity labels together with configurable search terms such as “synonym” and “translation” and consider the number and quality of the results that are returned. This has several benefits over using WordNet or a domain-specific dictionary:

- It will work with jargon, slang, and other words not likely to be in WordNet.

- It will work with non-English labels.

- It will always be up-to-date with the way words are currently being used by real people.

1.3 Adaptations made for the evaluation

Because the OAEI competition does not consider instance matches in the results, these types of matches are removed from alignment after the algorithm has completed (so that they can still be used by the metrics) but before the final results are stored.

Due to the nature of the OAEI test sets (particularly the benchmark test set), the syntactic (lexical) metric is set to only consider exact lexical matches rather than a Levenstein distance greater than some threshold.

1.4 Link to the system and parameters file

The source code for MapSSS can be downloaded at https://github.com/mcheatham/MapSSS. No parameters file is required.

1.5 Link to the set of provided alignments

The results produced by the system are also available at https://github.com/mcheatham/MapSSS.

2 Results

MapSSS has only been tested extensively on the benchmark test set, but it can produce results for the conference set as well. There are currently problems related to the memory requirements of some methods within Jena (a Java library used to read in and manipulate owl files) that prevent the algorithm from providing results for the anatomy test.

2.1 benchmark

Using the same results format as the ASMOV authors last year, the MapSSS results on the 12 different difficulty levels within the benchmark test set are shown in the table below.

Difficulty	Precision	Recall	F1
0	1.00	1.00	1.00
1	.74	.75	.74
2	.74	.73	.74
3	.89	.84	.86

4	.99	.95	.97
5	.99	.87	.93
6	.99	.74	.85
7	.98	.63	.77
8	.97	.42	.58
9	.70	.13	.22
10	.28	.02	.04
3xx	.93	.69	.80

It is encouraging that MapSSS has high precision on many of the difficulty levels. It should be straightforward to improve the precision on levels 1 and 2 by adding a semantic metric capable of understanding language translations. The algorithm's recall is not nearly so good currently. Plans to improve this are discussed in Section 3.2.

2.2 conferences

During the development of this preliminary version of MapSSS, testing was done primarily using the benchmarks test set. However, the system is also capable of producing results for the conferences test set. The f-measures for each possible pair of ontologies in the test set are shown in the table below. Performance varies widely among the different ontology pairs. Future work on MapSSS will involve analyzing these results in detail to understand and mitigate this variability.

	confOf	**conf.**	**edas**	**ekaw**	**iasted**	**sigkdd**
cmt	.29	.36	.72	.50	.57	.76
confOf		.50	.55	.48	.57	.47
conf.			.53	.44	.36	.48
edas				.35	.42	.67
ekaw					.42	.70
iasted						.69

3 General comments

3.1 Comments on the results

The precision of MapSSS is very reasonable in most cases, but the recall leaves something to be desired in many of the test cases. The next section outlines several possible improvements to the algorithm that are likely to increase its recall. We hope

to compare the baseline results presented here with those from an improved version of the algorithm next year.

3.2 Discussions on the way to improve the proposed system

We have several thoughts on improving the performance of MapSSS:

- The semantic metric needs to be implemented. This should improve the algorithm's recall in several cases, particularly when there is not much class hierarchy or there are few relationships between classes.

- After the current algorithm has found all of the matches it can, a second pass can be made that will relax some of the exactness constraints of the metrics, particularly the structural metric. For instance, instead of requiring exact edge matches, a category-oriented approach could be used that would treat e.g. all cardinality restrictions as equivalent. While it is not likely this will improve results on the synthetic OAEI benchmarks, it may improve the recall on the real-world test sets.

- Again after the current algorithm has found all of the matches it can, the structural metric can be run again but rather than looking for exact graph matches, if one graph is a subgraph of the other, ontology reasoning algorithms can be employed in a directed manner to see if the missing links can be inferred.

- The memory and computation requirements of the algorithm can be improved by more careful implementation of the metrics and by spawning off new processes when recursing on a newly discovered match and then merging the results back into the main processing thread.

3.3 Comments on the OAEI 2011 procedure

The SEALS platform is very helpful because it allows testing of algorithms from any computer, without the need to copy the test files to each system. Participants can also always be assured of having the most up-to-date test files.

The two biggest issues were:

- It was not clear from the tutorial that the source files need to be in a subdirectory named after the package name.

- Validation consistently fails because the build file does not seem to properly put the names of the libraries into the descriptor template.

3.4 Comments on the OAEI 2011 test cases

The OAEI test cases are invaluable in allowing ontology alignment algorithms to be compared against one another (and previous versions of the algorithm) on a consistent set of problems.

It would be helpful if there was a SEALS test set that used more of the OWL vocabulary, in order to fully test algorithms that consider that.

As a minor note, Jena, which is a Java library commonly used to read in and manipulate ontologies, does not consider some of the benchmark files (the benchmarksI set that was used in 2010) to be valid. This is because the xml namespace of some entities is an empty string (xmlns = “”).

4 Conclusion

MapSSS is very preliminary work towards an OWL alignment algorithm that uses expensive reasoning techniques sparingly to achieve quality alignments. The algorithm uses a syntactic, structural, and (in the future) semantic metric to determine matching nodes, and takes advantage of the locality present in most ontologies by recursing whenever it finds a match. The current version has reasonable precision but low recall, which the future work outlined here will hopefully improve.