CroMatcher - Results for OAEI 2013

Marko Gulić[1], Boris Vrdoljak[2]

[1] Faculty of Maritime Studies, Rijeka, Croatia
marko.gulic@pfri.hr
[2] Faculty of Electrical Engineering and Computing, Zagreb, Croatia
boris.vrdoljak@fer.hr

Abstract. CroMatcher is an ontology matching system based on terminological and structural matchers. The most important part of the system is automated weighted aggregation of correspondences produced by using different basic ontology matchers. This is the first year CroMatcher has been involved in the OAEI campaign. The results obtained this year will certainly help in finding and resolving shortcomings in the system before the next campaign.

1 Presentation of the system

CroMatcher is an automatic ontology matching system for determining correspondences between entities of two different ontologies. There are several terminological and structural basic matchers in CroMatcher. The system is based on a weighted aggregation that automatically determines the importance of each basic matcher according to the produced correspondences. As this is the first time the CroMatcher has taken part in the OAEI campaign, CroMatcher is fully prepared only for benchmark test set.

1.1 State, purpose, general statement

CroMatcher is a system that executes several basic matchers and then aggregates the results obtained by these matchers. The system does not use any external resource. After the execution of terminological basic matchers, the automatic weighted aggregation is executed. The results of certain terminological basic matcher are included into the common results depending on their importance. The importance of certain basic matcher is determined automatically within weighted aggregation. Then, the several iterative structural matchers are executed (e.g. if the child entities are similar, the parent entities are similar too). To find correspondences with structural matchers, the common results of terminological matchers are used. After the execution of structural basic matchers, the automatic weighted aggregation is executed too. At the end of matching process, the weighted aggregation is executed for the terminological and structural common results. Finally, the method of final alignment (choosing the relevant correspondences between entities of two ontologies) is executed. This method iteratively takes the best correspondences between two certain entities into the final alignment. Each entity can be related just to one entity of other ontology.

1.2 Specific techniques used

In this section, the main components of the CroMatcher will be described in details. The workflow and the main components of the system can be seen in the Fig. 1. The CroMatcher consists of the following components:

1. **Data extraction from ontologies** - the information of every entity is extracted from given ontologies. After extraction of all data about certain entity, all textual data is normalized by tokenizing into set of tokens, and removing stop words.

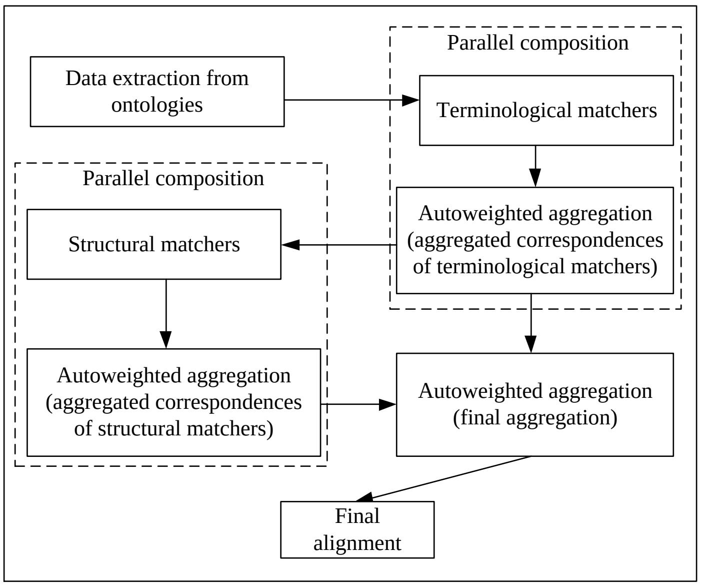
Parallel composition
Data extraction from ontologies
Terminological matchers
Parallel composition
Structural matchers
Autoweighted aggregation (aggregated correspondences of terminological matchers)
Autoweighted aggregation (aggregated correspondences of structural matchers)
Autoweighted aggregation (final aggregation)
Final alignment

Fig. 1. The workflow and the main components of the Cromatcher

2. **Terminological matchers:**

- Matcher that compares ID and annotations' text of two entities (classes or properties) with the bi(tri)gram matcher (tests how many bi(tri)grams, i.e. (substrings of length 2, 3) are the same within two names, e.g. FTP and FTPServer have 2 bigrams - FT and TP) [1]
- Matcher that compares only label (or entity's ID if the entity does not have label) of two entities (classes or properties) with the bi(tri)gram matcher
- Matcher that compares textual profiles of two entities with TF/IDF [2] and cosine similarity [3]. A profile of class entity contains annotations of actual class entity (and all sub classes) and annotations of every property whose domain is actual class. A profile of property entity contains annotations of actual property entity and all sub properties.

- Matcher that compares individuals of two entities with TF/IDF and cosine similarity. An individual of class entity contains individual values of actual class entity and individual values of all subclasses. An individual of property entity contains individual values of its range class entities.
- Matcher that compares extra individuals of two entities with TF/IDF and cosine similarity. An extra individual of class entity contains individual values of first super class of actual class entity. An extra individual of property entity contains individual values of its domain and range class entities.
- Matcher that compares some general data about the entities. A general data of class entity contains number of object (data) properties, number of restrictions and number of sub (super) class entities. A general data of property entity contains number of sub (super) property entities, number of domain class entities. More similar the general data, there is the greater correspondence between entities.

3. **Structural matchers:**

- Matcher that compares the similarity between super entities (classes or properties) of currently compared entities. If the super entities are similar, compared entities are similar too. The matcher is executed iteratively and it ends when the correspondence value of compared entities stops changing. In each step, the new correspondence value of compared entities is calculated by summing 50% of the previous similarity value and 50% of the similarity value between super entities.
- Matcher that compares the similarity between sub entities (classes or properties) of currently compared entities. If the sub entities are similar, the compared entities are similar too. The matcher is executed iteratively and it ends when the correspondence value of compared entities stops changing. In each step, the new correspondence value of compared entities is calculated by summing 50% of the previous similarity value and 50% of the similarity value between sub entities.
- Matcher that compares the similarity between properties (and its range classes) that have the currently compared classes as their domain. A part of matcher for similarity between properties compares domain classes of properties.
- Matcher that compares the similarity between range classes of currently compared properties.

4. **Autoweighted aggregation for parallel composition of basic matchers:**

After the execution of terminological and structural matchers, the results of these matchers have to be aggregated together. In our system, we used a parallel composition of matchers for integration of multiple matchers. The main problem in parallel composition is how to aggregate the results obtained by every basic matcher. Weighted aggregation is one of the methods for aggregation of matchers [4]. This method determines a weighted sum of similarity values of the basic matchers and needs relative weights which should correspond to the expected importance of the basic matchers. The problem is how to determine the importance of every basic matcher. Our automatic Autoweight method proposed in [5] automatically defines the importance of various basic matchers in order to improve overall performance of the matching system. In this method, the importance of certain basic matcher is specified by determining the importance of individual best correspondences (greatest correspondences between two entities in both directions of mapping, as those correspondences are the most relevant) within the results obtained by that matcher. The importance of a certain correspondence found within the results of a basic matcher is higher when the same correspondence is found within a smaller number of other basic matchers. The method that finds the same correspondences as all other methods does not provide any new significant information for the matching process.

5. **Process of final alignment:**

At the end, the selection of relevant correspondences, for inclusion in the final alignment, is executed iteratively. The final alignment includes only the greatest correspondences between entity_{1i} (first ontology) and entity_{2j} (second ontology). A correspondence between entity_{1i} and entity_{2j} is the greatest correspondence only if it has the greatest value among all correspondences in which the entity_{1i} (or entity_{2j}) is included. Threshold for these greatest correspondences is set to 0.15. We consider that this threshold is sufficient because the final alignment included only those correspondences that are the greatest for both compared entities.

1.3 Link to the system and parameters file

A system can be downloaded from the http://www.seals-project.eu (tool identifier: e0fe95d5-943e-4652-bc53-5b36b712c9cb, version: 1.0).

2 Results

In this section, the evaluation results of CroMatcher matching system executed on the SEALS platform are presented.

2.1 Benchmark

In OAEI 2013, benchmark includes one blind test (biblio). In Table 1 the result obtained by running the CroMatcher ontology system can be seen.

Test set	Recall	Precision	F-Measure	Time (s)
Benchmark	0.82	0.95	0.88	1114

Table 1. CroMatcher result for benchmark track

2.2 Anatomy, conferences, multifarm, library, large biomedical ontologies and instance matching

This is the first year CroMatcher has been involved in the OAEI campaign and the focus was on benchmark track. Therefore, the system had problems with other tracks because we did not manage to test the system for other tracks before the evaluation due the lack of time. This year, the ontology matching system had to finish matching the anatomy ontologies within 10 hours, and our system has not finished even after 30 hours therefore we need to speed up the system before the next OAEI campaign. In the conference track, our system was partially evaluated because it could not process several ontologies. In the multifarm, library, large biomedical ontologies, our system gave an “OutOfMemory” exception so we need to solve that problem too before the next evaluation. Regarding the instance matching, we did not participate in this track.

3 General comments

As we stated before, this is the first time the CroMatcher system participates in the OAEI campaign. We are very pleased that our ontology matching system was evaluated on the SEALS platform because this way we could compare our system with existing systems. There are many different test cases and we think that these test cases will help us to improve our system in the future.

3.1 Comments on the results

Our system shows great results in benchmark track. Considering the fact that the benchmark track contains the largest number of ontologies in which the different parts are missing, we can conclude that our system performs well but only while matching small ontologies like these in the benchmark track. While matching big ontologies (thousands of entities), our system is quite slow and it cannot handle big ontologies yet.

3.2 Discussions on the way to improve the proposed system

We will have to find faster measure than TF/IDF to compare different documents of entities. Also, we will have to store the data about the entities in a separate file instead in the java objects in order to reduce the usage of memory in the system.

4 Conclusion

The CroMatcher ontology matching system and its results of evaluation on different OAEI track were presented in this paper. The evaluation results show that CroMatcher successfully matches small ontologies but it has problems dealing with ontologies that have a large number of entities. We will try to solve this problem and prepare the system to be competitive in all OAEI tracks next year.

References

1. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, 2007.

2. Salton, G., McGill, M.H.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
3. Baeza-Yates, R., Ribeiro-Neto B.: Modern Information Retrieval. Addison-Wesley, Boston (1999)
4. Do, H., Rahm, E.: COMA - a system for flexible combination of schema matching approaches. In Proc. 28th International Conference on VLDB, pages 610-621, 2002.
5. Gulić, M., Magdalenić, I., Vrdoljak, B.: Automatically Specifying Parallel Composition of Matchers in Ontology Matching Process. In: Barriocanal, E. G., Cebeci, Z., Okur, M. C., Öztürk, A. (eds.) MTSR 2011. Communications in Computer and Information Science, vol. 240, pp. 22-33. Springer, Berlin Heidelberg (2011)