TOMATO : results of the 2022 OAEI evaluation campaign

Philippe Roussille[1], Olivier Teste[2]

[-1] *Institut de Recherche en Informatique de Toulouse, Toulouse, France*

Abstract

This paper presents the results obtained by the TOMATO system in the OAEI 2022 evaluation campaign. We describe here the results in the Conference track. We report a general discussion on the results and on the future improvements of the system.

1. Presentation

1.1. Overview

TOMATO (***TO**olkit for **MAT**ching **O**ntologies*) is heavily inspired by POMAP++[1]. It is a pairwise ontology matching system; ie. matching couples of ontologies together. We employ mostly Machine Learning approaches based on element similarities[2] (using mainly string-based similarities).

The workflow of TOMATO is depicted in Figure 1. The workflow being built upon several steps starting with loading the OWL ontologies and the reference RDF aligned files.

Tomato can be used in two different ways :

1. in a **learning mode**, where it can generate a model given two ontologies and their reference matching : $train(O_1, O_2, reference(O_1, O_2)) = model(O_1, O_2)$
2. in a **matching mode**, where a previously generated model is used against two ontologies to produce an alignment : $matching(O_1, O_2, model(O_1, O_2)) = alignment(O_1, O_2)$

In both cases, we can choose to apply a global matching strategy, using all the elements from both ontologies given their types and producing a single model ; or a set of local matching strategies, depending on the data, in where we treat subsets of the elements from both ontologies (given a specific characteristic of the data) to produce one model per subset.

1.2. Matching steps

1.2.1. Loading ontologies

For its base operation, TOMATO parses and loads a given set of two input ontologies. We rely mostly on the `owlready2` Python library to populate our internal structure, which contains all

Proceedings of the 17th International Workshop on Ontology Matching

Philippe.Roussille@irit.fr (P. Roussille); Olivier.Teste@irit.fr (O. Teste)

CEUR Workshop Proceedings (CEUR-WS.org)

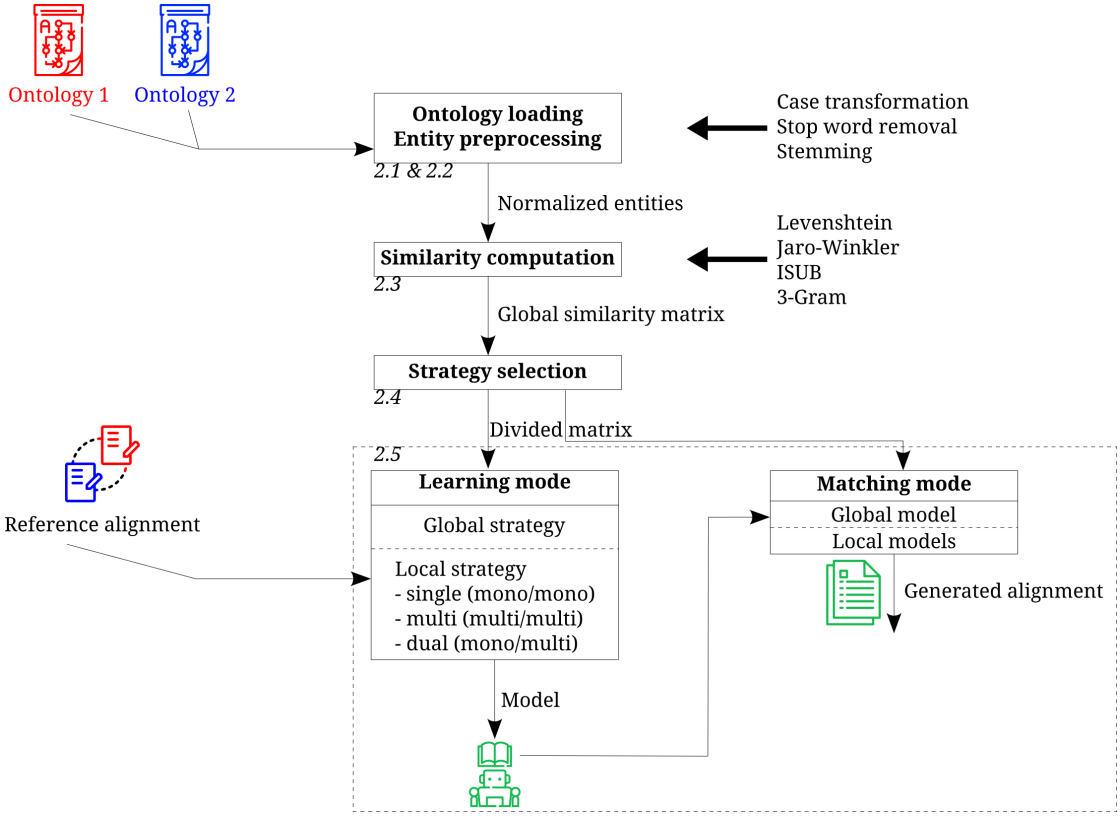

Figure 1: System workflow

the elements indexed by their respective names (classes, data properties and object properties), and their relations.

The name of an element is either using its label (if found) or, for a fallback approach, the last component of its URI.

For the relations, we chose to store all the relationships between classes.

1.2.2. Ontology preprocessing

Since string similarity measures are case-sensitive, symbol-sensitive and very dependent on what words are being parsed, we first need to apply multiple filters in order to have data that can be properly analysed.

Here are the steps we chose to keep :

- CamelCase words are converted to snake_case, which allows us to separate words ;
- snake_case and non-alphanumeric symbols are replaced by spaces ;
- we perform an English stemming process as the one proposed by Porter[3] ;
- we then remove all stop words (such as "a", "who", "me", etc.) using WordNet approach to have only the concept words ;

- we stem the words for the similarity measures that support them.

1.2.3. Similarity score computation

Once the preprocessing steps are complete, the input ontologies are quite ready to be processed by the level matcher.

The entities are then paired : we compute all possible combination of class x class, data property x data property, and object property x object property. For each combination, we compute the similarity of the couple through an all-against-all strategy, which stands for measuring the similarity score (using a combination of Levenshtein, JaroWinkler, ISUB, and 3-Gram similarity measures). This step leaves us with a vector of scores for each possible couple, coordinates being the different similarity measures used.

1.2.4. Strategy selection

To ensure a better result, we apply a local strategy based on what kind of entity we are matching. We defined three possible situations :

- both entities contains only a single word ("mono"-term) ;
- both entities contains more than one word ("multi"-term) ;
- entities contains a different term count ("dual"-term).

This strategy is somewhat naive, but it would serve as a starting point for our next exploration.

1.2.5. Train and match

Training a model In order to find the best measures that can indicate a potential match, we use a classifier from the `scikit-learn` Python module (`GridSearchCV`). The train/test split ratio is 0.4 : we use 60% of our data to train the model, and 40% to test, doing ten splits of cross-validation for each of the three models.

We feed the similarity matrix with the reference data (normalized as "0" if no match and "1" if there is a match). We pass the similarity matrix to the SVM machine-learning algorithm, and we save the model trained. Given our localized strategy, we end up with three different models, one for each situation (mono, multi, dual).

Computing a match In order to compute a match, we build a similarity matrix for all the couples we are interested in. The matrix is then fed through our classifier using the previously trained model. The classifier then outputs the expected alignment.

1.3. Adaptations made for OAEI

For our participation in OAEI 2022, we decided to test against both strategies (local and global) so we could tune our models. As such, we participated using only our local strategies. We included our comparison in the next section.

2. Results

2.1. Method

TOMATO runs on a Ubuntu server enviroment, with 16 GiB of RAM and a Intel Xeon Gold 6244 CPU at 3.6GHz. We can submit jobs to it through the web interface at
`http://tomato.irit.fr/match`

2.2. Comparison global and local strategies

The global strategy consists in applying a single model for all possible couples of conferences. This model is therefore built by finding a global optimum between the similarity measures corresponding to the features, and the model weights

The Local strategy consists in applying a model depending on the entity word count (mono, multi or dual) as per section 1.2.4.

Ontology 1	Ontology 2	Global strategy			Local strategy		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
cmt	**conference**	0.04	0.07	0.05	0.5	0.2	0.29
cmt	**confof**	0.05	0.13	0.07	0.57	0.25	0.35
cmt	**edas**	0.07	0.23	0.1	0.88	0.54	0.67
cmt	**ekaw**	0.1	0.27	0.15	0.83	0.45	0.59
cmt	**iasted**	0.02	0.25	0.04	0.67	1	0.8
cmt	**sigkdd**	0.01	0.08	0.02	0.75	0.5	0.6
conference	**confof**	0.04	0.2	0.06	0.89	0.53	0.67
conference	**edas**	0.09	0.29	0.14	0.64	0.41	0.5
conference	**ekaw**	0.04	0.12	0.06	0.58	0.28	0.38
conference	**iasted**	0.05	0.14	0.08	0.57	0.29	0.38
conference	**sigkdd**	0.08	0.27	0.12	0.83	0.33	0.48
confof	**edas**	0.05	0.21	0.09	0.56	0.47	0.51
confof	**ekaw**	0.02	0.05	0.03	0.58	0.35	0.44
confof	**iasted**	0.03	0.11	0.04	0.57	0.44	0.5
confof	**sigkdd**	0.04	0.29	0.07	1	0.57	0.73
edas	**ekaw**	0.01	0.04	0.02	0.67	0.26	0.38
edas	**iasted**	0.08	0.05	0.06	0.83	0.26	0.4
edas	**sigkdd**	0.01	0.07	0.01	1	0.47	0.64
ekaw	**iasted**	0.01	0.1	0.02	0.56	0.5	0.53
ekaw	**sigkdd**	0.08	0.36	0.13	1	0.55	0.71
iasted	**sigkdd**	0.13	0.33	0.18	0.73	0.53	0.62
all ontologies		0.04	0.16	0.07	0.7	0.4	0.51

Table 1
Results ran locally on our models between global and local

Applying local models helps to smooth the averaging effects of the averages made by using a single model for all the ontologies. While our strategy is very simple, it helps showing us that selecting a different model based on some feature from the data gives us better results as shown in table 1.

2.3. OAEI results

2.3.1. Ranking

Matcher	Precision	Recall	F1-measure
LogMap	0.76	0.56	0.64
GraphMatcher	0.75	0.55	0.63
SEBMatcher	0.79	0.48	0.6
ATMatcher	0.69	0.51	0.59
ALIN	0.82	0.44	0.57
edna	0.74	0.45	0.56
LogMapLt	0.68	0.47	0.56
AMD	0.82	0.41	0.55
LSMatch	0.83	0.41	0.55
StringEquiv	0.76	0.41	0.53
KGMatcher+	0.83	0.38	0.52
ALIOn	0.66	0.19	0.3
TOMATO	**0.09**	**0.6**	**0.16**
Matcha	0.37	0.07	0.12

Table 2
TOMATO (in bold) ranked against the other matchers, sorted by F1-measure

While our system has the highest recall (0.6) of the list, we came second to last in terms of F-Measure as table 2 shows. This is due to our low precision score (9%), and shows the limits of using only a simple string-matching strategy.

2.3.2. Specifics

	Precision	F-measure	Recall
Sharp	0.09	0.16	0.63
Discrete	0.08	0.15	0.74
Continuous	0.08	0.15	0.73

Table 3
TOMATO details for the conference track

The OAEI 2022 results for conference cover three situations, as described in the track details :

- *sharp*, where the reference alignements tested against TOMATO are all 1.0;
- uncertain situation, where the confidence value of a match has been set equal to the percentage of a group of people who agreed with the match in question (this uncertain version is based on reference alignment labeled `ra1-M3`):
  - *discrete*, which considers that there is a correct match if the reference alignment has a confidence value of 0.5 or greater ; and a fully incorrect match if the confidence value is less than 0.5;

– *continuous*, which penalizes TOMATO more if it misses a match on which most people agree than if it misses a more controversial match : if the reference alignement gives that e_1 and e_2 match with a 0.85 confidence, and if a matching algorithm gives that match a confidence of 0.40, then that is counted as 0.85 * 0.40 = 0.34 of a true positive and 0.85 – 0.40 = 0.45 of a false negative.

In all three cases, while we have a good recall, our very low precision score is detrimental to our F-measure, and highlights once more the limits of our simplistic approach.

3. Conclusions

This paper presented the TOMATO matcher, a matcher that produces pairwise ontology alignements using a Machine Learning approach mixed with a strategy selection depending on the data used.

While our strategy is very simple, it is sufficient to show that our hypothesis is correct : using a local strategy depending on the data is better than a global single strategy.

In future versions, we intend to study other Machine Learning algorithms and different similarity measures to improve the quality of the generated alignments. We will also study the structure of the ontologies in depth to increase our performances, and include other measures to strengthen our results.

References

[1] A. Laadhar, F. Ghozzi, I. Megdiche, F. Ravat, O. Teste, F. Gargouri, POMap++ results for OAEI 2019: fully automated machine learning approach for ontology matching, in: 14th International Workshop on Ontology Matching co-located with the International Semantic Web Conference (OM@ISWC 2019), Auckland, New Zealand, 2019, pp. 169–174. URL: https://hal.archives-ouvertes.fr/hal-02942337.

[2] M. Cheatham, P. Hitzler, String similarity metrics for ontology alignment, in: H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. Noy, C. Welty, K. Janowicz (Eds.), The Semantic Web – ISWC 2013, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 294–309.

[3] M. F. Porter, An algorithm for suffix stripping, Program 14 (1980) 130–137. URL: https://doi.org/10.1108/eb046814. doi:10.1108/eb046814.