# YAM++ : A multi-strategy based approach for Ontology matching task

Duy Hoa Ngo, Zohra Bellahsene

## ▶ To cite this version:

Duy Hoa Ngo, Zohra Bellahsene. YAM++ : A multi-strategy based approach for Ontology matching task. Knowledge Engineering and Knowledge Management, Oct 2012, Galway City, Ireland. pp.5. lirmm-00720639

HAL Id: lirmm-00720639

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00720639

Submitted on 25 Jul 2012

# YAM++ : A multi-strategy based approach for Ontology matching task

DuyHoa Ngo, Zohra Bellahsene
{firstname.name@lirmm.fr}

Université Montpellier 2, INRIA, LIRMM
Montpellier - France

**Abstract.** In this paper, we present the capability of our ontology matching tool YAM++. We show that YAM++ is able to discover mappings between entities of given two ontologies by using machine learning approach. Besides, we also demonstrate that if the training data are not available, YAM++ can discover mappings by using information retrieval techniques. Finally, we show that YAM++ is able to deal with multi-lingual ontologies matching problem.

## 1 Introduction

There are many challenges in ontology matching task. A matcher tool can be seen as a combination of three sub-matchers such as: Element level matcher, Structural level matcher and Semantical matcher. Generally, element level matcher discovers mappings by comparing annotation (i.e., labels, comments) of entities. It may use many different similarity metrics to handle the high terminological heterogeneity of ontologies. A challenging issue here is how to combine different metrics effectively. Additionally, if labels of entities in ontologies are represented by different languages, the matching process is even more difficult. Structural level matcher discovers mappings of entities based on analyzing their structural information. However, according to [9], most of them don't perform well when the structures of ontologies are different. Moreover, structural matcher is error-prone, since it strongly depends on initial mappings provided by element level matcher. Semantical matcher is mainly used to refine candidate mappings. It exploits the semantic constraints between entities in ontologies in order to remove inconsistent mappings. It is a NP-complete problem [5] to find the global optimization results (i.e., the minimum set of inconsistent mappings). To handle these challenges, we propose our solution as follows:

- If labels of entities are written by different languages, we use a multi lingual translator to translate labels from other languages into English.
- We use machine learning based approach to combine different similarity metrics at element level matcher. In case we don't have training data, we propose a similarity metrics based on information retrieval techniques.
- We use a graph matching, in particular similarity propagation method, which is known as the most stable method dealing with structural information to discover additional mappings.
- In terms of semantic refinement, we use the Global Optimal Diagnosis method [5].

The rest of the paper is organized as follows. In Section 2, we present an overview of YAM++ system. Section 3 contains the demonstration scenarios. In section 4, we summarize our contributions.

## 2 YAM++ Overview

Fig. 1 depicts the main components of the YAM++ system. YAM++ discovers mappings between two input ontologies by two matchers: element level matcher and structural level matcher. The combination of the mappings resulting from element level and structural level are then revised by the semantical matcher in order to remove inconsistent mappings.

- At element level, input ontologies are processed in order to extract annotation information for every entity. Based on these information, similarity score between entities are computed by different terminological metrics. Here, similarity metrics can work independently or can be combined by combination methods in order to produce mappings at element level. Currently, YAM++ supports machine learning based combination methods such as Decision Tree, SVM, NaiveBayes[1], etc. In that case, the training data are provided by the user or are taken from knowledge base (KB) resources. Otherwise, by default, YAM++ performs our proposed matching method based on information retrieval technique.
- At structural level, input ontologies are parsed and transformed into graph data structure. Then, YAM++ takes the mappings resulting from element level as initial mappings to run a similarity propagation process. The propagation algorithm here is inspired from the well known Similarity Flooding algorithm [11]. See [8] for more detail about our extension of similarity propagation method.
- In semantical checking module, we make use of global constraint optimization method proposed in Alcomox tool[2]
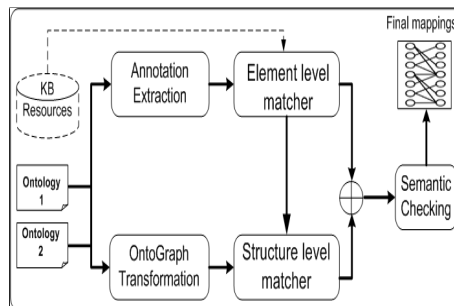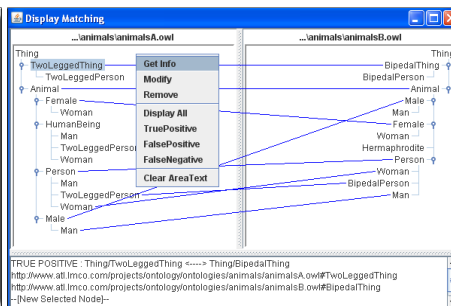


Fig. 1: YAM++ architecture



Fig. 2: Graphical User Interface

The resulting mappings of the matching process are displayed in graphical user interface (Fig. 2). The user can judge a mapping as correct or not according to his/her

---

[1] http://www.cs.waikato.ac.nz/ml/weka/

[2] http://web.informatik.uni-mannheim.de/alcomo/

knowledge of ontologies'domain. The user can also modify, remove incorrect mappings or add new mappings with the help of command operations shown in YAM++'s menu.

## 3 Demonstration Scenarios

YAM++ has been implemented in Java, offering a GUI to select different configuration options and display the matching results. In this demo, we will show the following capabilities of YAM++: (i) matching with machine learning method, (ii) matching with information retrieval method, and (iii) matching with multi-lingual ontologies.

In order to evaluate the matching quality of YAM++, we compute three standard evaluation metrics(i.e., H-mean precision, recall and Fmeasure) on two data sets **Conference**[3] and **Multifarm**[4] in order to compare the matching quality of our system with other participants of OAEI campaign[5].

### 3.1 Matching with machine learning method

In the first scenario, we assume that the user has several gold standard data sets, which consist of two ontologies and a corresponding alignment provided by experts of the domain. The user may think that he/she can study some matching patterns from the existing data sets to discover new mappings from new matching scenario with to-be-matched ontologies. Obviously, manually finding mappings is not applicable with the big size of ontologies. Therefore, the user would like to use the existing data as training data to train a machine learning model. Then, the learning model will automatically examine every pair of entities from to-be-matched ontologies and classify them into match or not.

Based on this idea, YAM++ provides different kinds of similarity metrics, which can be used to represent different features of each pair of entities from two to-be-matched ontologies. For example, from the YAM++'s control panel, the user can select string-based metrics such as Levenstein, QGrams[6]; linguistic-based metrics on Wordnet[7] such as Lin, Wu-Palmer, etc. [10]. The matching process could be decomposed into three steps: (i) Learning phase. The user will select a set of similarity metrics, a gold standard dataset and a machine learning model. (ii) YAM++ creates training data and performs a training process. (iii) Classification phase. YAM++ generates a classification, which is used to classify and produce mappings between the input ontologies. Furthermore, if the user chooses the option of running structural method from YAM++'s control panel for the next step, these mappings will be passed to input of the similarity propagation process. Finally, the mapping results will be shown in the display for user's judgment. More details about the similarity metrics and their combinations are described in [6, 7].

---

[3] http://oaei.ontologymatching.org/2011/conference/index.html

[4] http://web.informatik.uni-mannheim.de/multifarm/

[5] http://oaei.ontologymatching.org/2011.5/

[6] http://secondstring.sourceforge.net/

[7] http://wordnet.princeton.edu/

| Matcher | Prec. | $F_{0.5}$Meas. | Rec. | Prec. | $F_1$Meas. | Rec. | Prec. | $F_2$Meas. | Rec. |
|---|---|---|---|---|---|---|---|---|---|
| YAM++ | .8 | .73 | .53 | .78 | **.65** | .56 | .78 | .59 | .56 |
| CODI | .74 | .7 | .57 | .74 | .64 | .57 | .74 | .6 | .57 |
| LogMap | .85 | **.75** | .5 | .84 | .63 | .5 | .84 | .54 | .5 |
| AgrMaker | .8 | .69 | .44 | .65 | .62 | .59 | .58 | **.61** | .62 |
| *BaseLine₂* | **.79** | .7 | **.47** | **.79** | **.59** | **.47** | **.79** | **.51** | **.47** |
| MaasMtch | .83 | .69 | .42 | .83 | .56 | .42 | .83 | .47 | .42 |
| *BaseLine₁* | **.8** | **.68** | **.43** | **.8** | **.56** | **.43** | **.8** | **.47** | **.43** |
| CSA | .61 | .58 | .47 | .5 | .55 | .6 | .5 | .58 | .6 |
| CIDER | .67 | .61 | .44 | .64 | .53 | .45 | .38 | .48 | .51 |
| MapSSS | .55 | .53 | .47 | .55 | .51 | .47 | .55 | .48 | .47 |
| Lily | .48 | .42 | .27 | .36 | .41 | .47 | .37 | .45 | .47 |
| AROMA | .35 | .37 | .46 | .35 | .4 | .46 | .35 | .43 | .46 |
| Optima | .25 | .28 | .57 | .25 | .35 | .57 | .25 | .45 | .57 |
| MapPSO | .28 | .25 | .17 | .21 | .23 | .25 | .12 | .26 | .36 |
| LDOA | .1 | .12 | .56 | .1 | .17 | .56 | .1 | .29 | .56 |
| MapEVO | .27 | .08 | .02 | .15 | .04 | .02 | .02 | .02 | .02 |

Fig. 3: Comparison result on Conference 2011 track [3]

Fig. 3 shows the comparison result of YAM++ with other participants on the Conference track in OAEI 2011 campaign[8]. This was the first time we participate in the OAEI competition. At this time, YAM++ stayed in **Top2** among all participants. Especially, in terms of $F_1$ measure, it achieved the **best** matching tool title.

### 3.2 Matching with information retrieval method

In this second scenario, we assume that related gold standard data sets are not available. In that case, the method of using machine learning model is not applicable. Instead, YAM++ provides matching methods based on information retrieval techniques. In particular, YAM++ applies information retrieval technique on annotation information of entities to determine amount of informativeness of tokens within the input ontologies. Then, an extended metric of Tversky's theory [4] has been developed to compute similarity between entities' labels. Similar to the first scenario, the user can select similarity propagation to discover more mappings by exploiting structural information of entities.

Fig. 4 shows the comparison result of YAM++ with other participants on the Conference track in OAEI 2011.5 campaign[9]. This was the second time we participate to the OAEI competition with non learning YAM++ system. At this time, YAM++ obtained the **best** matching result and dominated all other participants.

### 3.3 Matching with multi-lingual ontologies

In the last scenario, we show the ability of YAM++ to work with multi-lingual ontologies matching. When the user provides two to-be-matched ontologies, YAM++ read annotations of entities in order to determine which language is used in each ontology.

---

| Matcher | Threshold | Precision | F0.5-measure | F1-measure | F2-measure | Recall |
|---------|-----------|-----------|--------------|------------|------------|--------|
| YAM++ | 0 | 0.8 | 0.78 | 0.74 | 0.71 | 0.69 |
| LogMap | 0 | 0.82 | 0.75 | 0.66 | 0.59 | 0.55 |
| CODI | 0 | 0.74 | 0.7 | 0.64 | 0.6 | 0.57 |
| Hertuda | 0 | 0.79 | 0.7 | 0.6 | 0.53 | 0.49 |
| WeSeE | 0.33 | 0.71 | 0.65 | 0.59 | 0.53 | 0.5 |
| Baseline2 | 0 | 0.79 | 0.7 | 0.59 | 0.51 | 0.47 |
| LogMapLt | 0 | 0.73 | 0.67 | 0.59 | 0.53 | 0.5 |
| GOMMA | 0 | 0.84 | 0.71 | 0.58 | 0.49 | 0.44 |
| AUTOMSv2 | 0 | 0.79 | 0.68 | 0.56 | 0.47 | 0.43 |
| Baseline1 | 0 | 0.8 | 0.68 | 0.56 | 0.47 | 0.43 |
| MaasMtch | 0.89 | 0.74 | 0.64 | 0.54 | 0.46 | 0.42 |
| MapSSS | 0 | 0.5 | 0.5 | 0.5 | 0.51 | 0.51 |
| MapPSO | 0.67 | 0.1 | 0.08 | 0.07 | 0.06 | 0.05 |
| MapEvo | 0.82 | 0.04 | 0.03 | 0.03 | 0.02 | 0.02 |

Fig. 4: Comparison result on Conference 2011.5 track [1]

Once the languages are defined, YAM++ uses Microsoft Bing Translator tool[10] to translate all labels from other languages to English. After that, YAM++ discovers mappings between entities based on their translated labels by using proposed information retrieval methods. The returned mappings are passed as input to similarity propagation process to discover more mappings.

| Matching system | Different ontologies (type i) | | | | Same ontologies (type ii) | | | |
|-----------------|------|-----------|--------|-----------|------|-----------|--------|-----------|
| | Size | Precision | Recall | F-measure | Size | Precision | Recall | F-measure |
| YAM++ | 1838 | 0.54 | 0.39 | 0.45 | 5838 | 0.93 | 0.48 | 0.63 |
| AUTOMSv2 | 746 | 0.63 | 0.25 | 0.36 | 1379 | 0.92 | 0.16 | 0.27 |
| WeSeE | 4211 | 0.24 | 0.39 | 0.29 | 5407 | 0.76 | 0.36 | 0.49 |
| CIDER | 737 | 0.42 | 0.12 | 0.19 | 1090 | 0.66 | 0.06 | 0.12 |
| MapSSS | 1273 | 0.16 | 0.08 | 0.10 | 6008 | 0.97 | 0.51 | 0.67 |
| LogMap | 335 | 0.36 | 0.05 | 0.09 | 400 | 0.61 | 0.02 | 0.04 |
| CODI | 345 | 0.34 | 0.04 | 0.08 | 7041 | 0.83 | 0.51 | 0.63 |
| MaasMtch | 15939 | 0.04 | 0.28 | 0.08 | 11529 | 0.23 | 0.23 | 0.23 |
| LogMapLt | 417 | 0.26 | 0.04 | 0.07 | 387 | 0.56 | 0.02 | 0.04 |
| MapPSO | 7991 | 0.02 | 0.06 | 0.03 | 6325 | 0.07 | 0.04 | 0.05 |
| CSA | 8482 | 0.02 | 0.07 | 0.03 | 8348 | 0.49 | 0.36 | 0.42 |
| MapEVO | 4731 | 0.01 | 0.01 | 0.01 | 3560 | 0.05 | 0.01 | 0.02 |

Fig. 5: Comparison on Multifarm 2011.5 track [2]

Fig. 5 shows the comparison result between YAM++ and other participants of OAEI 2011.5 campaign on Multifarm data sets. There are two types of evaluation. In the first type, all matching tools deal with different ontologies with different languages. In this evaluation, YAM++ achieved the **best** matching quality (Fmeasure = 0.45). In

---

[10] http://www.microsofttranslator.com/

the second type, all tools discover mappings of the same ontologies but translated in different languages. In this evaluation, YAM++ obtained the **second** position among all participants.

## 4  Conclusion

In this paper, we present YAM++ - an ontology matching tool, which supports: (i) discovering alignment of ontologies by machine learning approaches; (ii) discovering alignment of ontologies by generic methods without using learning techniques; (iii) discovering alignment of ontologies represented in different languages. Moreover, our tool produced high matching quality results on Benchmark, Conference and Multifarm tracks in comparison with other participants on OAEI 2011 and OAEI 2011.5 campaigns. The running tool can be found at http://www2.lirmm.fr/∼dngo/.

## References

[1] http://oaei.ontologymatching.org/2011.5/results/conference/.

[2] http://oaei.ontologymatching.org/2011.5/results/multifarm/.

[3] http://oaei.ontologymatching.org/2011/results/conference/.

[4] A.Tversky. Features of similarity. *Psychological Review*, pages 327–352, 1977.

[5] C.Meilicke. Alignment incoherence in ontology matching. In *PhD Thesis, University of Mannheim, Chair of Artificial Intelligence*, 2011.

[6] DH.Ngo, Z.Bellahsene, and R.Coletta. A flexible system for ontology matching. In *CAISE LBIP*, pages 79–94, 2011.

[7] DH.Ngo, Z.Bellahsene, and R.Coletta. A generic approach for combining linguistic and context profile metrics in ontology matching. In *ODBASE*, pages 800–807, 2011.

[8] DH.Ngo, Z.Bellahsene, and R.Coletta. Yam++ results for oaei 2011. In *OM*, 2011.

[9] J.Euzenat and P.Shvaiko. *Ontology matching*. Springer-Verlag, 2007.

[10] Q.Budanitsky and G.Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Comput. Linguist.*, pages 13–47, 2006.

[11] S.Melnik, H.Garcia-Molina, and E.Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.