YAM++ results for OAEI 2013

Duy Hoa Ngo, Zohra Bellahsene

YAM++ – Results for OAEI 2013

DuyHoa Ngo, Zohra Bellahsene

University Montpellier 2, LIRMM
{duyhoa.ngo, bella}@lirmm.fr

Abstract. In this paper, we briefly present the new YAM++ 2013 version and its results on OAEI 2013 campaign. The most interesting aspect of the new version of YAM++ is that it produces high matching quality results for large scale ontology matching tasks with good runtime performance on normal hardware configuration like PC or laptop without using any powerful server.

1 Presentation of the system

YAM++ - (not) Yet Another Matcher is a flexible and self-configuring ontology matching system for discovering semantic correspondences between entities (i.e., classes, object properties and data properties) of ontologies. This new version YAM++ 2013 has a significant improvement from the previous versions [6, 5] in terms of both effectiveness and efficiency, especially for very large scale tasks. The YAM++'s general architecture is not changed much. However, most of its algorithms have been updated/added by the new effective ones. For example, we have implemented a **disk-based** method for storing the temporary information of the input ontology during the indexing process in order to save main memor space. Consequently, the new version YAM++ 2013 has improved both the matching quality and time performance in large scale ontology matching tasks. This year, YAM++ participates in **six tracks** including **Benchmark**, **Conference**, **Multifarm**, **Library**, **Anatomy** and **Large Biomedical Ontologies** tracks. However, due to limitation of time and person, we have not upgrade YAM++ to participate to **Interactive matching evaluation** and **Instance Matching** tasks.

1.1 State, purpose, general statement

In YAM++ approach all useful information of entities such as terminological, structural or contextual, semantic and extensional are exploited. For each type of extracted information, a corresponding matching module has been implemented in order to discover as many as possible candidate mappings.

The major drawback of the previous version **YAM++ 2012** [5], despite the fact that it achieved good results and high ranking at almost tracks, is very low time performance, especially for the **Large Biomedical Ontologies** tracks. After carefully studying this issue, we realize that our algorithms for pre-processing and indexing the input ontologies lead to a high complexity of $O(n^2)$, where n is the size of the ontology. Additionally, the semantic verification component did not work well for the very large scale ontology matching task.

In the current version **YAM++ 2013**, the flaws mentioned above have been significantly fixed. Firstly, we have revised our algorithms for pre-processing and indexing the input ontologies and now they are with $O(\|n\| + \|v\|)$ complexity, where n and v are the number of nodes and edges of a Directed Acyclic Graph transformed from the ontology. Moreover, we have implemented a **disk-based** method for storing the temporary information of the input ontologes during the indexing process. This method allows us save a significant space of main memory. this makes possible run YAM++ with very large scale ontology matching in a personal computer with ordinary configuration (using 1G JVM only).

Secondly, we have introduced different inconsistent alignment patterns in order to detect as much as possible conflict set. Then, a new and fast approximate algorithm has been implemented to find the nearly optimization solution, which corresponds to the final consistent alignment.

1.2 Specific techniques used

In this section, we will briefly describe the workflow of YAM++ and its main components, which are shown in Fig.1.

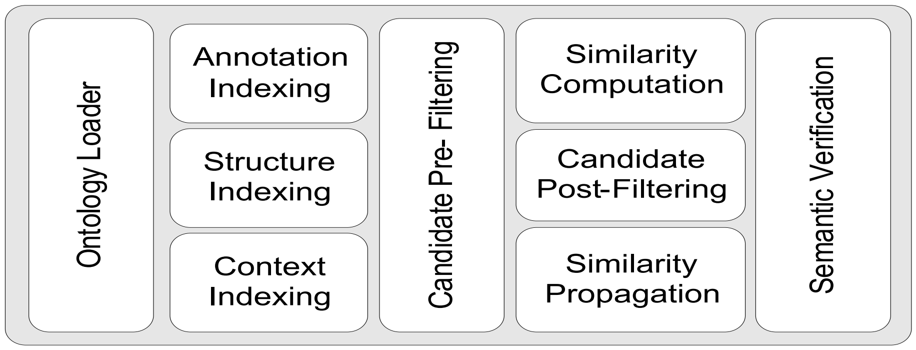


Fig. 1: Main components of YAM++ system

In YAM++ approach, a generic workflow for a given ontology matching scenario is as follows.

1. Input ontologies are loaded and parsed by a **Ontology Loader** component;
2. Information of entities in ontologies are indexed by the **Annotation Indexing**, the **Structure Indexing** and **Context Indexing** components;
3. **Candidates Pre-Filtering** component filters out all possible pairs of entities from the input ontologies, whose descriptions are highly similar;
4. The candidate mappings are then passed into **Similarity Computation** component, which includes: (i) the **Terminological Matcher** component that produces a set of mappings by comparing the annotations of entities; (ii) the **Instance-based Matcher** component that supplements new mappings through shared instances between ontologies and (iii) the **Contextual Matcher**, which is used to compute the similarity value of a pair of entities by comparing their context profiles. In YAM++,

the matching results of the **Terminological Matcher**, the **Contextual Matcher** and the **Instance-based Matcher** are combined to have a unique set of mappings. We call them element level matching result.

5. The **Similarity Propagation** component then enhances element level matching result by exploiting structural information of entities; We call the result of this phase structure level matching result.
6. The **Candidate Post-Filtering** component is used to combine and select the potential candidate mappings from element and structure level results.
7. Finally, the **Semantic Verification** component refines those mappings in order to eliminate the inconsistent ones.

Let us now to present the specific features of each component.

Ontology Loader To read and parse input ontologies, YAM++ uses OWLAPI open source library. In addition, YAM++ makes use of (i) Pellet[1] - an OWL 2 Reasoner in order to discover hidden relations between entities in small ontology and (ii) ELK[2] reasoner for large ontology. In this phase, the whole ontology is loaded in the main memory.

Annotation Indexing In this component, all annotations information of entities such as ID, labels and comments are extracted. The languages used for representing annotations are considered. In the case where input ontology use different languages to describe the annotations of entities, a multilingual translator (**Microsoft Bing**) is used to translate those annotations to English. Those annotations are then normalized by tokenizing into set of tokens, removing stop words, and stemming. Next, the resulting tokens are indexed in a table for future use.

Structure Indexing In this component, the main structure information such as IS-A and PAR-OF hierarchies of ontology are stored. In particular, YAM++ assigns a compressed bitset values for every entity of the ontology. Through the bitset values of each entity, YAM++ can fast and easily gets its ancestors, descendants, etc. A benefit of this method is to easily access to the structure information of ontology and minimize memory for storing it. After this step, the loaded ontology can be released to save main memory.

Context Indexing In this component, we define a context profile of an entity as a set of three text corpora: (i) Entity Description includes annotation of the entity itself; (ii) Ancestor Description comprises the descriptions of its ancestor and (iii) Descendant Description comprises the descriptions of its descendant. Indexing those corpora is We performed by **Lucene** indexing engine.

Candidates Pre-Filtering The aim of this component is to reduce the computational space for a given scenario, especially for the large scale ontology matching tasks. In YAM++, two filters have been designed for the purpose of performing matching process efficiently.

[1] http://clarkparsia.com/pellet/
[2] http://www.cs.ox.ac.uk/isg/tools/ELK/

– A **Description Filter** is a search-based filter, which filters out the candidate mappings before computing the real similarity values between the description of entities. Firstly, the descriptions of all entities in the bigger size ontology are indexed by **Lucene** search engine. For each entity in the smaller size ontology, three multiple terms queries corresponding to three description included in its context profile will be performed. The **top-K** algorithm based on ranking score of those queries is used to select the most similar entities.
– A **Label Filter** is used to fast detect candidate mappings, where the labels of entities in each candidate mapping are similar or differ in maximum two tokens. The intuition is that if two labels of two entities differ by more than three tokens, any string-based method will produce a low similarity score value. Then, these entities are highly unmatched.

Similarity Computation The three matcher described in this component are the same as in the **YAM++ 2012** version. For more detail, we refer readers to our papers: **Terminological Matcher** [7], **Instance-based Matcher** [6]. A slight modification at the **Contextual Matcher** is that we use algorithm described in [8] for small ontology matching, whereas we use the **Lucene** ranking score for large scale ontology matching.

Similarity Propagation This component is similar to the **Structural Matcher** component described in YAM++ 2012 version. It contains two similarity propagation methods namely Similarity Propagation and Confidence Propagation.

– The **Similarity Propagation** method is a graph matching method, which inherits the main features of the well-known **Similarity Flooding** algorithm [2]. The only difference is about transforming an ontology to a directed labeled graph. This matcher is not changed from the first YAM++ version to the current version. Therefore, for saving space, we refer to section **Similarity Flooding** of [6] for more details.
– The **Confidence Propagation** method principle is as follows. Assume $\langle a_1, b_1, \equiv, c_1 \rangle$ and $\langle a_2, b_2, \equiv, c_2 \rangle$ are two initial mappings, which are maybe discovered by the element level matcher (i.e., the terminological matcher or instance-based matcher). If a_1 and b_1 are ancestors of a_2 and b_2 respectively, then after running confidence propagation, we have $\langle a_1, b_1, \equiv, c_1 + c_2 \rangle$ and $\langle a_2, b_2, \equiv, c_2 + c_1 \rangle$. Note that, confidence values are propagated only among collection of initial mappings.

In YAM++, the aim of the **Similarity Propagation** method is discovering new mappings by exploiting as much as possible the structural information of entities. This method is used for a small scale ontology matching task, where the total number of entities in each ontology is smaller than 1000. In contrary, the **Confidence Propagation** method supports a **Semantic Verification** component to eliminate inconsistent mappings. This method is mainly used in a large scale ontology matching scenario.

Candidates Post-Filtering The aim of the **Mappings Combination and Selection** component is to produce a unique set of mappings from the matching results obtained by the terminological matcher, instance-based matcher and structural matcher. In this component, a **Dynamic Weighted Aggregation** method have been implemented. Given an ontology matching scenario, it automatically computes a weight value for each matcher and establishes a threshold value for selecting the best candidate mappings. The main idea of this method can be seen in [6] for more details.

Semantic Verification After running the similarity or confidence propagation on overall candidate mappings, the final achieved similarity values reach a certain stability. Based on those values, YAM++ is able to remove inconsistent mappings with more certainty. There are two main steps in the **Semantic Verification** component such as (i) identifying inconsistent mappings, and (ii) eliminating inconsistent mappings.

In order to identify inconsistencies, several semantic conflict patterns have been designed in YAM++ as follows (see [4] for more detail):

- Two mappings $\langle a_1, b_1 \rangle$ and $\langle a_2, b_2 \rangle$ are crisscross conflict if a_1 is an ancestor of a_2 in ontology O_1 and b_2 is an ancestor of b_1 in ontology O_2.
- Two mappings $\langle a_1, b_1 \rangle$ and $\langle a_2, b_2 \rangle$ are disjointness subsumption conflict if a_1 is an ancestor of a_2 in ontology O_1 and b_2 disjoints with b_1 in ontology O_2 and vice versa.
- A property-property mapping $\langle p_1, p_2 \rangle$ is inconsistent with respect to alignment A if $\{Doms(p_1) \times Doms(p_2)\} \cap A = \emptyset$ and $\{Rans(p_1) \times Rans(p_2)\} \cap A = \emptyset$ then (p_1, p_2), where $Doms(p)$ and $Rans(p)$ return a set of domains and ranges of property p.
- Two mappings $\langle a, b_1 \rangle$ and $\langle a, b_2 \rangle$ are duplicated conflict if the cardinality matching is 1:1 (for a small scale ontology matching scenario) or the semantic similarity $SemSim(b_1, b_2)$ is less than a threshold value θ (for a large scale matching with cardinality 1:m).

Two methods, i.e., complete and approximate diagnosis are used in order to eliminate inconsistent mappings. We use complete version Alcomo [3] for small scale. In term of approximate version for large scale, we transform this task into a **Maximum Weighted Vertex Cover** problem. A modification of **Clarkson** algorithm [1], which is a **Greedy** approach. The idea of this method is that it iteratively removes the mapping with the smallest cost, which is computed by a ratio of its current confidence value to number of its conflicts.

1.3 Adaptations made for the evaluation

Before running the matching process, YAM++ analyzes the input ontologies and adapts itself to the matching task. In particular, if the annotations of entities in input ontologies are described by different languages, YAM++ automatically translates them in English. If the number of entities in input ontologies is smaller than 1000, YAM++ is switched to small scale matching regime, otherwise, it runs with large scale matching regime. The main difference between the two regimes lies in the **Similarity Propagation** and **Semantic Verification** components as we discussed above.

1.4 Link to the system and parameters file

A SEALS client wrapper for YAM++ system and the parameter files can be download at: http://www2.lirmm.fr/d̃ngo/YAMplusplus2013.zip. See the instructions in tutorial from SEALS platform[3] to test our system.

[3] http://oaei.ontologymatching.org/2013/seals-eval.html

1.5 Link to the set of provided alignments (in align format)

The results of all tracks can be downloaded at: http://www2.lirmm.fr/d̃ngo/ YAMplusplus2013Results.zip.

2 Results

In this section, we present the evaluation results obtained by running YAM++ with SEALS client with **Benchmark**, **Conference**, **Multifarm**, **Library**, **Anatomy** and **Large Biomedical Ontologies** tracks. All experiments are executed by YAM++ with SEALS client version 4.1 beta and JDK 1.6 on PC Intel 3.0 Pentium, 3Gb RAM, Window XP SP3.

2.1 Benchmark

In OAEI 2013, Benchmark includes 5 blind tests for both organizers and participants. Those tests are regeneration of the bibliography test set. Table 1 shows the avergae results of YAM++ running on the Benchmark dataset.

Test set	H-mean Precision	H-mean Recall	H-mean Fmeasure
Biblio	0.97	0.82	0.89

Table 1: YAM++ results on pre-test Benchmark track

2.2 Conference

Conference track now contains 16 ontologies from the same domain (conference organization) and each ontology must be matched against every other ontology. This track is an open+blind, so in the Table 2, we can only report our results with respect to the available reference alignments

Test set	H-mean Precision	H-mean Recall	H-mean Fmeasure
Conference ra1	0.80	0.69	0.74
Conference ra2	0.78	0.65	0.71

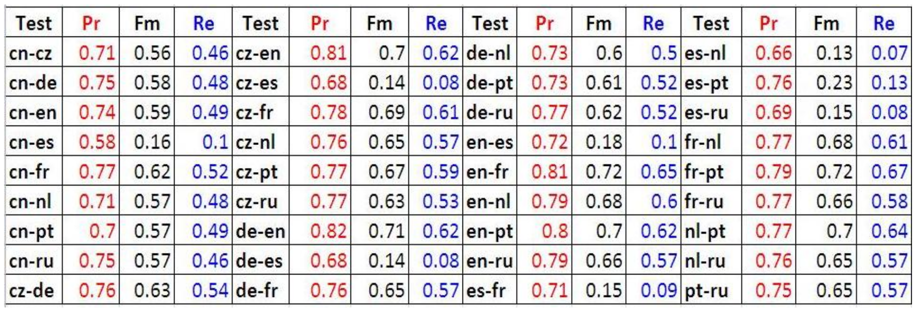
Table 2: YAM++ results on Conference track

2.3 MultiFarm

The goal of the MultiFarm track is to evaluate the ability of matcher systems to deal with multilingual ontologies. It is based on the OntoFarm dataset, where annotations of entities are represented in different languages such as: English (en), Chinese (cn), Czech (cz), Dutch (nl), French (fr), German (de), Portuguese (pt), Russian (ru) and Spanish (es). YAM++'s results are showed in the Fig. 2

2.4 Anatomy

The Anatomy track consists of finding an alignment between the Adult Mouse Anatomy (2744 classes) and a part of the NCI Thesaurus (3304 classes) describing the human anatomy. Table 3 shows the evaluation result and runtime of YAM++ on this track.



Test	Pr	Fm	Re	Test	Pr	Fm	Re	Test	Pr	Fm	Re	Test	Pr	Fm	Re
cn-cz	0.71	0.56	0.46	cz-en	0.81	0.7	0.62	de-nl	0.73	0.6	0.5	es-nl	0.66	0.13	0.07
cn-de	0.75	0.58	0.48	cz-es	0.68	0.14	0.08	de-pt	0.73	0.61	0.52	es-pt	0.76	0.23	0.13
cn-en	0.74	0.59	0.49	cz-fr	0.78	0.69	0.61	de-ru	0.77	0.62	0.52	es-ru	0.69	0.15	0.08
cn-es	0.58	0.16	0.1	cz-nl	0.76	0.65	0.57	en-es	0.72	0.18	0.1	fr-nl	0.77	0.68	0.61
cn-fr	0.77	0.62	0.52	cz-pt	0.77	0.67	0.59	en-fr	0.81	0.72	0.65	fr-pt	0.79	0.72	0.67
cn-nl	0.71	0.57	0.48	cz-ru	0.77	0.63	0.53	en-nl	0.79	0.68	0.6	fr-ru	0.77	0.66	0.58
cn-pt	0.7	0.57	0.49	de-en	0.82	0.71	0.62	en-pt	0.8	0.7	0.62	nl-pt	0.77	0.7	0.64
cn-ru	0.75	0.57	0.46	de-es	0.68	0.14	0.08	en-ru	0.79	0.66	0.57	nl-ru	0.76	0.65	0.57
cz-de	0.76	0.63	0.54	de-fr	0.76	0.65	0.57	es-fr	0.71	0.15	0.09	pt-ru	0.75	0.65	0.57

Fig. 2: YAM++ results on MultiFarm track

Test set	Precision	Recall	Fmeasure	Run times
Anatomy	0.944	0.869	0.905	62 (s)

Table 3: YAM++ results on Anatomy track

2.5 Library

The library track is a real-word task to match the STW (6575 classes) and the TheSoz (8376 classes) thesaurus. Table 4 shows the evaluation result and runtime of YAM++ against an existing reference alignment on this track.

Test set	Precision	Recall	Fmeasure	Run times
Library	0.692	0.808	0.745	411 (s)

Table 4: YAM++ results on Library track

2.6 Large Biomedical Ontologies

This track consists of finding alignments between the Foundational Model of Anatomy (FMA), SNOMED CT, and the National Cancer Institute Thesaurus (NCI). There are 9 sub tasks with different size of input ontologies, i.e., small fragment, large fragment and the whole ontologies. Table 5 shows the evaluation results and run times of YAM++ on those sub tasks.

3 General comments

This is the third time YAM++ participates to the OAEI campaign. We found that SEALS platform is a very valuable tool to compare the performance of our system with the others. Besides, we also found that OAEI tracks covers a wide range of heterogeneity in ontology matching task. They are very useful to help developers/researchers to develop their semantic matching system.

3.1 Comments on the results

The current version of YAM++ has shown a significant improvement both in terms of matching quality and runtime with respect to the previous version. In particular, the

Test set	Precision	Recall	Fmeasure	Run times
Small FMA - NCI	0.976	0.853	0.910	94 (s)
Whole FMA - NCI	0.899	0.846	0.872	366 (s)
Small FMA - SNOMED	0.982	0.729	0.836	100 (s)
Whole FMA - SNOMED	0.947	0.725	0.821	402 (s)
Small SNOMED - NCI	0.967	0.611	0.749	391 (s)
Whole SNOMED - NCI	0.881	0.601	0.714	713 (s)

Table 5: YAM++ results on Large Biomedical Ontologies track

H-mean Fmeasure value of all the very large scale dataset (i.e., Library, Biomedical ontologies) has been improved.

4 Conclusion

In this paper, we have presented our ontology matching system called YAM++ and its evaluation results on different tracks on OAEI 2013 campaign. The experimental results are promising and show that YAM++ is able to work effectively and efficiently with real-world ontology matching tasks. In near future, we continue improving the matching quality and efficiency of YAM++. Furthermore, we plan to deal with instance matching track also.

References

[1] Kenneth L. Clarkson. A modification of the greedy algorithm for vertex cover. *Information Processing Letters*, pages 23 – 25, 1983.

[2] Sergey Melnik el at. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.

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

[4] DuyHoa Ngo. Enhancing ontology matching by using machine learning, graph matching and information retrieval techniques. In *PhD.Thesis, University Montpellier II*, 2012.

[5] DuyHoa Ngo and Zohra Bellahsene. Yam++ results for oaei 2012. In *OM*, 2012.

[6] DuyHoa Ngo, Zohra Bellahsene, and Remi Coletta. Yam++ results for oaei 2011. In *OM*, 2011.

[7] DuyHoa Ngo, Zohra Bellahsene, and Konstantin Todorov. Opening the black box of ontology matching. In *ESWC*, pages 16–30, 2013.

[8] DuyHoa Ngo, Zohra Bellasene, and Remi Coletta. A generic approach for combining linguistic and context profile metrics in ontology matching. In *ODBASE Conference*, 2011.