Evaluating the Interaction between the different Matchers (or Strategies) in Ontology Matching Task

Duy Hoa Ngo, Zohra Bellahsene

Evaluating the Interaction between the different Matchers (or Strategies) in Ontology Matching Task

DuyHoa Ngo, Zohra Bellahsene

Université Montpellier 2, INRIA, LIRMM
161 rue Ada, 34095, Montpellier, France
{firstname.lastname@lirmm.fr}

Abstract. Ontology matching is a key solution to deal with the semantic heterogeneity problem in semantic web and data integration task. Generally, due to the high heterogeneity of ontologies, the combination of many methods are necessary to discover mappings. Then, an ontology matching tool can be seen as a set of several matching modules; each of them implementing a specific method (e.g., terminological based, structural based, semantic based, etc.). Moreover, many aspects should be taken into consideration. For example, what is the interaction between the all modules working together. In this paper, we study and evaluate with OAEI datasets two main issues such as: (i) global-based vs. local-based matching approach ; (ii) impact of input mappings on the quality of matching modules. In these experiments, we demonstrate that in element and structure based matchers, our proposed global based methods outperform the local ones and are more stable.

1 Introduction

There are many challenges in ontology matching task. The full picture of all challenges can be seen at [20]. In this paper, we focus only on matcher selection and combination problems, which are the fundamental and most difficult in developing of almost matching tools/systems. Generally, a matching tool/system can be seen as a combination of three matcher modules such as: Element based matcher, Structure based matcher and Semantic based matcher. Despite they exploit different features (i.e., terminological, structural and semantical) of entities of an ontology to discover mappings, they are not totally independent. Commonly, a structure based matcher takes as input the mappings resulting from element based matcher [8, 24, 2]; a semantic based matcher may take as input the mappings resulting of either element based matcher [10, 6] of structure based matcher or a combination of the mappings resulting of both element and structure based matcher [4, 7]. Challenges and difficulties can be arisen not only inside each module but also due to the interaction between them. Let's see some of them that we take into consideration as follows:

– Element based 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 difficult challenge here is how

to select the most appropriate similarity metrics or how to select the effective combination of different metrics in order to produce a high matching quality result. Additionally, if labels of entities in ontologies are represented by different languages, the matching process is even more difficult.

- Structure based matcher discovers mappings of entities based on analyzing their similar structural patterns, to where entities belong. An example of entity's structural pattern is their neighbors entities. Thus, two entities are considered as similar if their neighbors are highly similar. However, according to [3], almost methods of this type are not stable and don't improve the matching quality when the structures of ontologies are different. It is because a structural pattern may work with a pair of entities but might not work with another pair. Moreover, structural matcher is error-prone, since it strongly depends on initial mappings provided by element level matcher.
- Semantic based matcher is mainly used to refine candidate mappings [7, 6, 10]. It exploits the semantic constraints between entities in ontologies in order to discover conflict between candidate mappings. A question here is what mappings will be removed. To do that, in some tools [6, 10], the semantic module requires a confidence value for each candidate mapping. Then, it applies a global optimization method in order to find the minimal inconsistent set of mappings. Therefore, it is error-prone like structural methods because it also depends on the confidence values of the mappings obtained from previous steps.

Based on the analyse on these challenges, we remark that the matching quality of one matcher module strongly depends on the input mappings provided by another matcher preceding it in the matching process. Therefore, our first intuition is that if we want to obtain a high matching quality result of the whole process, each matcher module should provide high matching quality.

In [3], matching approaches are divided in local based and global based methods. Local based methods judge the similarity between entities by comparing one particular kind of features of these entities, whereas, global based methods do it not only by using different features but also the semantic context that entities belong to. Therefore, our second intuition is that the global based methods are more appropriate than local based methods in the ontology matching task. It is because entities in ontology strongly connect to each other through logical axioms and semantic relations [5].

According to our intuition discussed above, we propose a hypothesis for developing an ontology matching tool as follows: it should be compounded by several matcher modules (i.e., Element based, Structure based and Semantic based matchers) which in term are implementations of a global based matching methods. In order to verify our hypothesis, we design a YAM++ [1] system for performing different configurations and different matching strategies over different matching scenarios. In YAM++, we propose global based matching methods as follows:

- In Element based matcher, we propose an information retrieval based method to compute similarity score between labels of entities.

[1] YAM++ - (not) Yet Another Matcher for Ontology Matching task.

– In Structure based matcher, we propose Similarity Propagation (SP) method for ontology matching, which is an extension of Similarity Flooding method [16]. This method is a graph matching method and belongs to global based methods group.
– In Semantic based matcher, we use the Global Diagnosis Optimization method proposed in [14]

The rest of the paper is organized as follows. In Section 2, we present an overview of YAM++ architecture. Section 3 contains the experimental evaluation on different matching scenarios. In section 4, we summarize our contributions.

2 Overview of our Architecture

The main components of YAM++ system are depicted in the lower part of Fig. 1. To perform evaluation of the quality of different matching strategies, YAM++ requires matching scenarios as input (upper part of Fig. 1). A matching scenario consists of two ontologies (i.e., source and target ontologies) and a reference alignment provided by an expert of domain. Given a matching scenario, input ontologies are loaded, pre-processed and transformed into YAM++'s internal data structure for storing information over these ontologies (Loading/Pre-Processing/Transforming component).

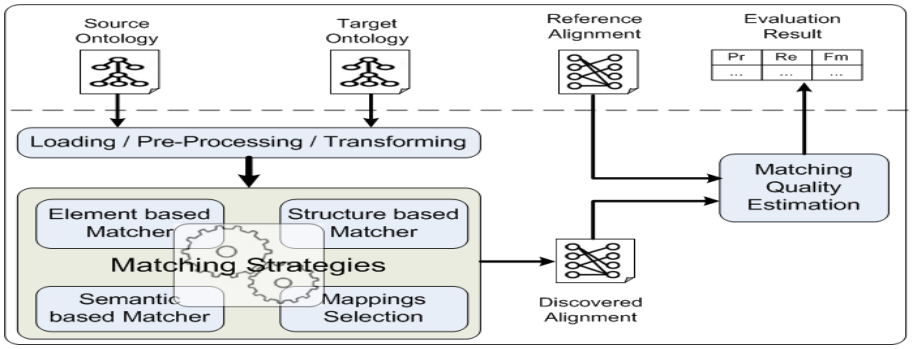

Fig. 1: System Architecture

In Fig. 1, a generic prototype of matching tool is displayed in the biggest rounded rectangle, which contains five modules: Element based matcher, Structure based matcher, Semantic based matcher, Mappings Selection and Matching strategies. The role of three matcher modules (i.e., Element based, Structure based, Semantic based) is to discover correct mappings or remove incorrect mappings according to features extracted from entities in input ontologies. Mappings Selection is a filter, which is aimed to select the most propable candidate mappings. In this paper, a matching strategy could be defined as the way the matcher and selection modules work together in the matching process to produce an alignment. Then, Matching Quality Estimation module evaluates the matching quality of given matching strategy by comparing discovered alignment with the reference alignment. Its outputs are three evaluation metric values corresponding to Precision (Pr), Recall (Re) and Fmeasure (Fm).

In YAM++ system, various matching methods inside three matcher modules (i.e., Element based, Structure based, Semantic based) and filtering methods used in Mappings Selection module have been implemented. More detail about matching and filter-

ing methods setup for each matching strategy will be discussed in each experiment in section 3.

3 Experimental Evaluation

In this section, we are going to present the setup of our experiments and discuss the experimental results obtained from them. More precisely, we will deal with two issues: (i) Comparison between our proposed global matching methods with other existing methods in the element based and structure based matchers; (ii) Impact of quality of input mappings on the Structure based and Semantic based matchers

All the experiments are executed with JRE 6.0 on Intel 3.0 Pentium, 3Gb of RAM, Window XP SP3. In these experiments, we use three standard evaluation metrics such as harmonic mean (H-mean) of precision, recall and f-measure to evaluate the matching quality of YAM++ on a set of tests.

$$H(p) = \frac{(\sum_{i=1}^{n} |C_i|)}{(\sum_{i=1}^{n} |A_i|)}; \quad H(r) = \frac{(\sum_{i=1}^{n} |C_i|)}{(\sum_{i=1}^{n} |R_i|)}; \quad H(fm) = \frac{2 * Hp * Hr}{(Hp + Hr)}$$

Here, assume that we have *n* tests. Let *i* indicates *i*th test; $|R_i|$ refers to the number of reference mappings provided by expert domain, $|A_i|$ is the total number of mappings discovered by a matching system and $|C_i|$ is the number of correct mappings.

3.1 Comparison of matching methods in Element based matcher module

The aim in this evaluation is to compare our proposed global method with other existing methods in terms of matching quality, in Element based matcher. To perform this experiment, we select Conference dataset including 21 test cases. The reason of this selection is that ontologies in dataset are moderate and are real world ontologies describing the same domain. Moreover, these ontologies are highly heterogeneous since they were developed by different people, hence, the same concept maybe labeled with different terminologies. Therefore, we assume that if a matcher obtain high quality of matching result on these tests, it would have good results on other real matching scenarios.

Local based methods at Element matcher. In YAM++, we have implemented more than 50 methods used in Element based matcher. We divide them into 3 main groups based on their characteristic of algorithm of computing similarity between strings. For the sake of saving space, the following representative methods will be used in this experiment:

- Levenstein[2], ISUB [21] - edit distance based methods which compute similarity of two strings based on number of edit operations.
- QGrams[3], TokLev - token based methods which split strings into set of tokens and then compare tokens by string based methods. Here, TokLev means tokens are compared by Levestein method. For more detail see [19].
- HybLinISUB, HybJCLev - hybrid based methods which split strings into set of tokens and then compare tokens by a combination method of a string based and linguistic based methods. Here, HybLinISUB means tokens are compared by combination of ISUB and Lin [12]; HybJCLev means tokens are compared by Levenstein and Jiang-Corath [9] methods. For more detail see [19].

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

[3] http://sourceforge.net/projects/simmetrics/

Global based methods at Element matcher. In our experiments, we have also implemented the following global based methods:

- Weighted Average with Local Confidence. Each local based method is assigned with a local confidence value. The local confidence values are used as weights in a weighted sum average function to compute the final similarity score between entities. More detail about local confidence values can be find in [2]. For short, we name it LC.
- Harmony based Adaptive similarity aggregation. Here, each local based method is assigned with a weight which is computed by the Harmony estimation algorithm [13]. Then, a weighted sum aggregation method is used to produce final similarity score between entities. For short, we name it HADAPT.
- Machine learning based approach combines all local based methods with some provided training data [17]. For short, we name it ML.
- Information Retrieval based method judges similarity between two entities by amount of overlap information of their labels. For short, we name it IR. IR splits all labels of entities into tokens and calculates the information content of each token in the whole ontology. Then, IR extends Tversky similarity measure [23] with weight of tokens to compute similarity score between labels of entities. Because our method compares similarity of two labels by using not only the sequence of characters themselves, but also their information content in ontology, we refer to our proposed method as a global based approach. We will illustrate this idea by examples in this experiment.

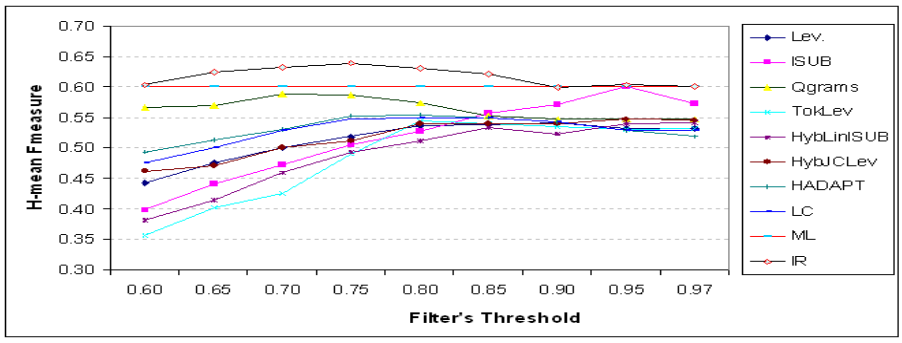

Fig. 2: Comparison of matching methods in Element based matcher module

The matching strategy is working as follows:

- Only Element based matcher and Mapping Selection modules are used.
- For each matching method (including local based and global based) used in element matcher, it computes similarity score between all pairs of entities of input ontologies.
- Mapping selection module selects candidate mappings according to the filter threshold value. A H-mean Fmeasure will be computed overall test cases in dataset.

Fig. 2 shows the diagram of comparison result between different methods used in element based matcher. Obviously, almost local based methods (except QGrams and ISUB) improve the Fmeasure when the threshold value in filter increases. When filter threshold gets high, it seems that only high similar or identical labels will be passed. Therefore, as can be seen in Fig. 2, local based methods closely converge to result of Identical method (≈ 0.55) when filter's threshold reaches to 0.95 and 0.97. It is the same trend for HADAPT and LC because they are linear combination of local based methods.

An interesting point here is about machine learning (ML) based combination method. In this experiment, in order to make sure the training data and test cases in Conference dataset are independent, we create different training data from OAEI Benchmark 2009[4] and I3CON[5] datasets. The result in the Fig. 2 is obtained by getting average 10 times running with 10 different training data. It shows that ML method does not require filter's threshold to select candidate mappings and returns a better matching quality than LC, HADAPT and other local based methods. For example, ML method discovers (cmt.owl#Co-author $\equiv$ conference.owl#Contribution_co-author) in ontologies `cmt.owl` and `conference.owl` respectively, whereas, local based methods return low similarity score between two labels (Levenstein(Co-author,Contribution_co-author) = 0.4; QGrams(Co-author,Contribution_co-author) = 0.6). It is understandable because ML does not use arithmetic combination functions like LC and HADAPT, instead, it extracts the combination rules on local methods from training data. There, ML can find many patterns in training data similar to the current example(e.g., (networkA.rdf#Office $\equiv$ networkB#OfficeSoftware), (russia1#payment $\equiv$ russia2#means_of_payment), etc.). However, ML method strongly depends on the training data. With different training data, different machine learning models will be generated and, therefore different matching results will be produced. For instance, with some training, ML can discover (cmt.owl#Co-author $\equiv$ conference.owl#Contribution_co-author), but with others, it cannot. Moreover, even in the same training data, this mapping is discovered by ML, but (cmt.owl#Document $\equiv$ conference.owl#Conference_document) cannot, even the latter mapping seems to look like the former one. Therefore, we have designed IR method, which is more stable than ML method.

In Fig. 2, our proposed IR method outperforms all other methods in the experiment. Let we explain the reason of this performance by an example with two entities: `cmt.owl`#Co-author and `conference.owl`#Contribution_co-author. After splitting and normalizing labels, we have 2 sets of tokens such as: {`coauthor`} and {`coauthor`, `contribution`}. Token `coauthor` appears in each input ontology only one time, whereas, token `contribution` appears 10 times among 60 concepts in ontology `conference.owl`. Therefore, the information content of token `contribution` is less than that of token `coauthor`. In particular, the normalized TFIDF weights of each token inside input ontologies are equal: $\{w_{coauthor} = 1.0\}$, $\{w_{coauthor} = 1.0$, $w_{contribution} = 0.34\}$. Two sets of tokens share only token `coauthor`, then the similarity computed by Tversky method is $\frac{1.0+1.0}{1.0+1.0+0.34} = 0.855$. Similarly, we have similarity between (`Document`,`Conference_document`) is `0.91`. In this pair, token `conference` appears 15 times in the `conference.owl` ontology. Therefore, we assume that this token brings small information in this ontology domain and, consequently, this pair of entities are likely matched.

This experiment shows that our proposed global based methods (ML and IR) produce better results than other methods do in element based matcher module.

[4] http://oaei.ontologymatching.org/2009/benchmarks

[5] http://www.atl.external.lmco.com/projects/ontology/i3con.html

3.2 Comparison of matching methods in Structural based matcher module

In this evaluation, we are going to compare the effectiveness of using our Similarity Propagation method (SP) [18] to other different existing structural methods. In particular, the following structural methods will be used in the comparison:

- ANCESTORS: two entities are similar if all or most of their ancestor entities are already similar [1].
- DESCENDANTS: two entities are similar if all or most of their descendant entities are already similar [1].
- LEAVES: two entities are similar if all or most of their leaf entities are already similar [1].
- ADJACENTS: two entities are similar if all or most of their adjacent entities (parents, children, siblings, domains, ranges) are already similar [11].
- ASCOPATH: two entities are similar if all or most of entities in the paths from the root to the entities in question are already similar [11].
- Descendant's Similarity Inheritance (DSIPATH): two entities are similar if the total contribution of entities in the paths from the root to them is higher than a specific threshold [22].
- Sibling's Similarity Contribution (SSC): two entities are similar if the total contribution of their sibling entities is higher than a specific threshold [22].

To perform this experiment, we have used Benchmark 2011 dataset including 103 test cases. These test cases are mainly considered for structural evaluation because of the following features: (i) Because entities don't have annotation(i.e., labels, comments) and their names are altered by random strings (no variation by naming convention or synonym words), therefore, the combination of different string based, linguistic based methods are not necessary. In this experiment, we can use only a simple string method to check whether two strings are identical or not. The interesting point here is that if two entities from two input ontologies have the same name, they are a correct mapping; (ii) In some tests, the structure of ontologies are not changed but a number of names are replaced by random strings. In other tests, not only names of entities are altered but also the ontology structure is changed (flatten, extension, etc.).

According to this observation, the matching strategy used in experiment is described as follows:

- Only 3 modules will be used: Element based matcher, Structure based matcher and Mapping selection.
- Element based matcher provides init mappings to structural based matcher. It uses Identical metric to compute similarity score between entities, which is equal to 1 if entities' names are the same and 0 otherwise.
- Each structure based matcher corresponding to each of selected structural metrics above produces a similarity matrix for all pairs of entities from the two input ontologies.
- We vary different threshold (0.01 - 0.9)to select mappings discovered by structural matcher. The mappings obtained by structural matcher are combined with mappings obtained by element based matcher to produce the set of candidate mappings. Then, a greedy selection method [15] is used to extract the final alignment.

Obviously, when the threshold varies from 0.6 to 0.9, the structural method lines in Fig. 3 seem to be converged with INIT-MAPPINGS line where H-mean Fmeasure = 0.68 (4463 correct mappings, 27 incorrect mappings, 4342 unfound). It means that the structural methods did not discover additional correct mappings or they discovered correct mappings, which already exist in input mappings. It is understandable because almost structural methods compute similarity between two entities by determining how much

overlap (e.g. Jaccard measure) of their structural patterns (i.e.m adjacent, ancestor, etc.). The higher filter threshold is, the lower possibility to discover new mappings is.

On the contrary, the matching quality of structural methods are significantly different when filter's threshold is set to small values. When the threshold is set to very small value (from 0.01 to 0.09), ASCOPATH and ANCESTORS provide low matching quality of results. It means that these methods discover many incorrect mappings. For example, when threshold is equal to 0.01, ACSOPATH discovers 90 (4733 - 4643) additional correct mappings but 453 (480 - 27) incorrect mappings in comparison with init mappings. It can be explained as follows. Due to observation of ontologies in Benchmark 2011 dataset, we see that the maximum depth and also maximum number of ancestors of an entity in the ontology hierarchy is equal to 5. Assume that two entities have only one common entity in their ancestors, then their similarity score at least is equal to $1/10 = 0.1$. If two entities don't have any common entity, then their similarity is equal to 0. Therefore, with threshold in range from 0.01 to 0.09, any pair of entities having at least one common ancestor will be assumed as matched. Since siblings entities have the same path and ancestors, they will have the same structural patterns. Therefore, many pairs of entities have the same similarity scores. Moreover, one entity may have many descendant entities so many pair of entities can be coupled, consequently, many incorrect mappings are produced.

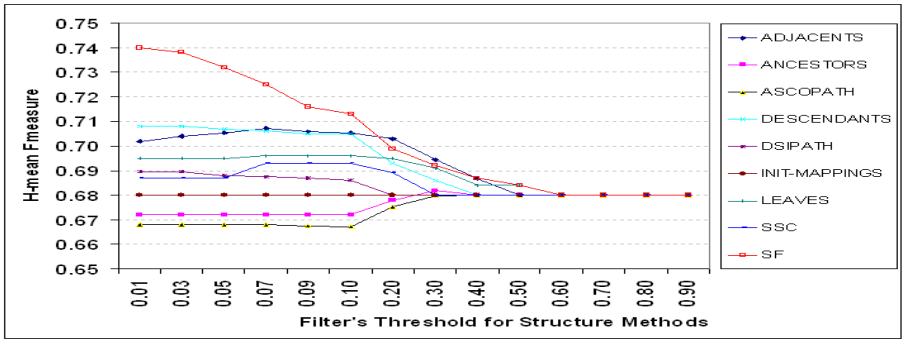

Fig. 3: Comparison on different matching methods in structure based matcher

Whereas, other methods such as DESCENDANTS, LEAVES, DSIPATH and SSC seem to work well with small thresholds. They discover more additional correct mappings than incorrect mappings and, consequently, they improve the quality of matching of the results. For example, with threshold is equal to 0.01, DESCENDANTS discovers 494 = (5137 - 4643) additional correct mappings and 175 = (202 -27) incorrect mappings in comparison with init mappings. Similar to ASCOPATH and ANCESTORS methods, with low threshold filter, many pairs of entities are passed. However, these methods clearly distinguish the structural patterns of entities. For instance, in DESCENDANTS and LEAVES, different entities have different sets of leaves/descendants; in DSIPATH and SSC, they use different contribution percentage of entities according to how much an entity is important to another [22]. Therefore, by running greedy selection, high percentage of selected mappings are correct.

Our proposed Similarity Propagation (SP) is different with these structural methods discussed above. Note that the similarity scores produced by SP is not the absolute but relative values due to normalized process at the end of each running iteration. SP propagates similarity values from one pair of entities to others, hence, if two entities have similarity score higher than 0, then they are somehow similar. Thus, with a low threshold filter, SP discovers more correct mappings than that with a high threshold value. Moreover, similarity score of a pair of entities depends on not only their current status but also on the status of the other pairs. The more neighbors with high similarity a pair of entities have, the higher possibility that they are matched. Therefore, SP well distinguishes correct and incorrect mappings by ranking similarity scores. That explains why SP outperform all other local based structural methods discussed above when the filter threshold is low. For example, when the threshold is equal to 0.01, SP discovers additional 1298 (5941 - 4643) correct mappings and 247 (274 - 27) incorrect mappings in comparison with init mappings. It shows that SP produces better matching quality result than other methods in the structure based matcher module.

3.3 Impact of noise input on structure based methods

In this experiment, we evaluate the behavior of different structure methods when we add noise data to input mappings. Here, we call a noise a pair of dissimilar entities but discovered as similar by element based matcher. It is important because in real scenario matching case, a matching method rarely produces 100% precision, consequently, it rarely provides input mappings without noise to structure methods. Intuitively, when noise data increase, the number of incorrect mappings increases whereas, the number of correct mappings decreases. Our assumption is that a stable method will produce less incorrect mappings than correct mappings. Therefore, we will study the changes of the number of correct and incorrect mappings discovered by each structure method. The evaluation strategy works as follows:

- At Element based matcher, we use Identical metric to produce initial mappings. In order to make noise, we add a number of random incorrect mappings to inputs, which is corresponding to N% of size of original init mappings. Here N = (0,10,..,100).
- At Structure based matcher, SP takes input mappings from Element based matcher and performs similarity propagation. According to the experiment in section 3.2, we select the best filter's threshold for each structure method. For example, $\theta_{SP} = 0.01$, $\theta_{DESCENDANTS} = 0.01$, $\theta_{ADJACENTS} = 0.07$, etc.
- For each running, we count the total number of correct mappings and the total number of incorrect mappings that a structure method produces overall 103 test cases in Benchmark 2011 dataset.

Fig. 4 shows the total number of correct and incorrect mappings produced by the structure methods for each time noise data are added to inputs. Generally, when more noise data are added, the number of correct mappings discovered by all the methods decreases ,whereas, the number of incorrect mappings discovered by almost methods increases except DSIPATH and SSC. Here, DSIPATH and SSC are unlike other local based structure methods in terms of interaction between entities in ontology. For example, the similarity of two entities computed by DSIPATH strongly depends on their similarity provided by input mappings and decreasingly depends on similarity of parents, grandparents, etc. Consider two entities of two input ontologies. If a noise appears at the

same level in their path to root, their similarity will be impacted by noise, otherwise, it will not. Therefore, the impact of a noise in discovering others mappings depends on the position of its entities in the hierarchies of input ontologies. Because noise data are created randomly, the impact of noise to produce incorrect mappings is unpredictable. Whereas, other structure methods use set operations (i.e. intersection, union), so there is no difference between parent and grandparent. When a noise appears in the set of ancestors or descendants of two entities, the noise will directly propagate errors to them. Therefore, obviously in Fig. 4, the number of incorrect mappings increases in almost structure methods.

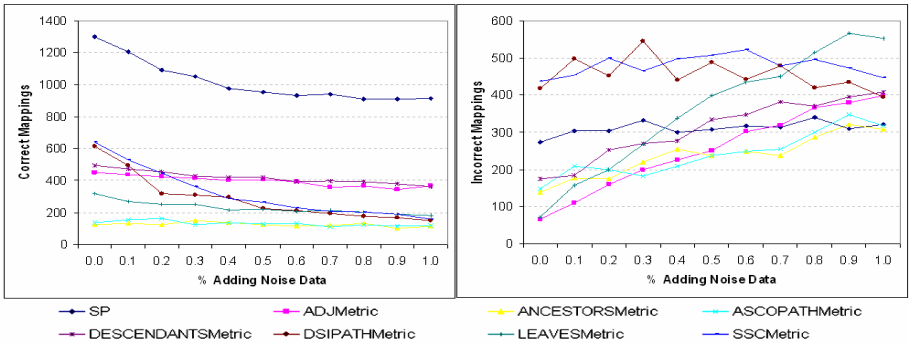

Fig. 4: Impact of noise input to structure based methods

This experiment also shows the dominant of using Similarity propagation over other structure methods. Let's see on the diagram representing the number of correct mappings discovered in Fig. 4. When the percentage of noisy data is 100%, SP still discovers 913 additional correct mappings in comparison with init mappings. Whereas, the maximum number of correct mappings discovered by the other methods is only 612 mappings when there is no noise added to the inputs. Moreover, in the next diagram in Fig. 4, from 0% to 100% of the noisy data, SP produces only 57 (321 - 274) additional incorrect mappings. Whereas, for example, LEAVES method produces 481 (553 - 72) more inccorect mappings. This feature is reasonable because SP takes all kinds of semantic relations of entities such as concept-concept, concept-property and property-property into account. These constraint relations will reduce the impact of the noisy data to produce mappings. That is why SP is known as a stability constraint method.

3.4 Impact of inputs quality on Structure and Semantic Matcher

In this evaluation, we are going to study the impact of matching quality of the input mappings to the matching quality of the structure and semantic matchers on Conference - a real world dataset. The matching strategy works as follows:

- At Element based matcher, we use a terminological method to produce initial mappings. According to the Fig. 2, we choose QGRams and ISUB matchers because they show different behavior when the element based filter's threshold changed.
- At Structure based matcher (SP) takes input from Element based matcher and perform similarity propagation. Whereas, Semantic based matcher (SM) refines input mappings in order to remove inconsistent ones.

– For each running, we evaluate the matching quality by comparing the discovered alignment with reference alignment.

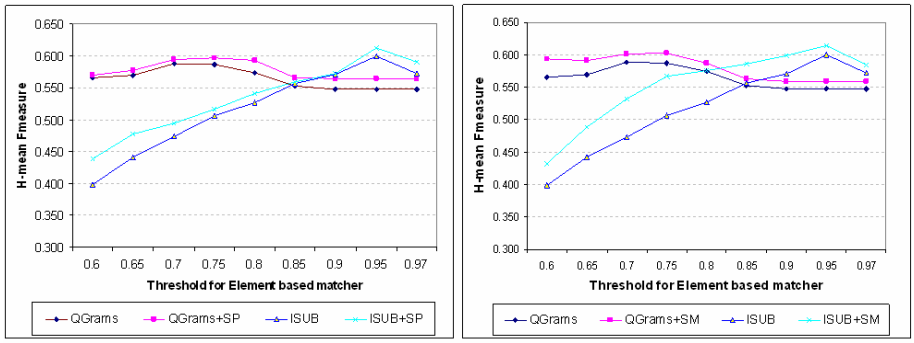

Fig. 5: Impact of inputs quality on Structure and Semantic Matchers

Fig. 5 shows the changes of matching quality obtained by SP and SM when init mappings changed. It seems that the two methods have the same behaviour. For example, when the threshold increases from 0.6 to 0.95, lines ISUB, ISUB+SP and ISUB+SM line go up. Next, when the threshold increase to 0.97, these lines go down. It happens similar to QGrams. Therefore, the intuition is that when the quality of input mappings improves, the matching quality of the results of SP and SM increases too and vice versa. According to this experiment, we may conclude that the better initial mappings provided to the structure and semantic matchers are, the better matching result will be obtained.

4 Conclusion

In this paper we have presented experiments and evaluations on two following aspects of ontology matching task: (i) advantage of using global based methods vs. local based methods in matching modules; (ii) impact of matching quality of the input mappings on the matching quality of matching modules. The experimental evaluations show that global based methods outperform local based methods in terms of matching quality and stability. Besides, the quality of input is very important because it directly impact to the result quality of matching modules.

5 Acknowledgments

In OAEI[6] 2011 campaign, YAM++ were winner in Conference track and second position in Benchmark track. In this first participation, YAM++ used machine learning method in element based matcher, similarity propagation method in structure based matcher and simple semantic verification method. In the second attending to campaign (OAEI 2011.5), YAM++ replaced machine learning method by new information retrieval method in element based matcher. Additionally, YAM++ used Microsoft Bing translator to translate labels from a given language to English in Multifarm track. In semantic based matcher, YAM++ reused Alcomox [14] tool to remove inconsistent

[6] http://oaei.ontologymatching.org

mappings. In the OAEI 2011.5 campaign, YAM++ achieved the first position in both Conference and Multifarm tracks. In Benchmark track, with two first datasets, YAM++ stayed in top 5 best matching tools.

References

[1] R. Dieng and S. Hug. In *Proceedings of ECAI 1998*, pages 341–345, 1998.
[2] Isabel F. Cruz et al. Using agreementmaker to align ontologies for oaei 2010. In *OM*, 2010.
[3] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
[4] Fausto Giunchiglia and Pavel Shvaiko et al. S-match: an algorithm and an implementation of semantic matching. In *In Proceedings of ESWS*, pages 61–75, 2004.
[5] Nicola Guarino. Formal ontology and information systems. pages 3–15. IOS Press, 1998.
[6] Jakob Huber, Timo Sztyler, Jan Nößner, and Christian Meilicke. Codi: Combinatorial optimization for data integration: results for oaei 2011. In *OM*, 2011.
[7] Yves R. Jean-Mary and Mansur R. Kabuka. Asmov: Results for oaei 2008. In *OM*, 2008.
[8] Ningsheng Jian, Wei Hu, Gong Cheng, and Yuzhong Qu. Falconao: Aligning ontologies with falcon. In *Integrating Ontologies*, 2005.
[9] Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, 1997.
[10] E. Jiménez-Ruiz, A. Morant, and B. Cuenca Grau. LogMap results for OAEI 2011. In *OM*, 2011.
[11] Bach Thanh Le, Rose Dieng-Kuntz, and Fabien Gandon. On ontology matching problems. In *ICEIS (4)*, pages 236–243, 2004.
[12] Dekang Lin. An information-theoretic definition of similarity. In *ICML Conference*, pages 296–304, 1998.
[13] Ming Mao, Yefei Peng, and Michael Spring. A harmony based adaptive ontology mapping approach. In *SWWS*, pages 336–342, 2008.
[14] Christian Meilicke. Alignment incoherence in ontology matching. In *Thesis*.
[15] Christian Meilicke and Heiner Stuckenschmidt. Analyzing mapping extraction approaches. In *OM'07*, pages –1–1, 2007.
[16] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.
[17] DuyHoa Ngo, Zohra Bellahsene, and Remi Coletta. A flexible system for ontology matching. In *Caise 2011 LBIP*, pages 79–94, 2011.
[18] DuyHoa Ngo, Zohra Bellahsene, and Remi Coletta. Yam++ results for oaei 2011. In *OM*, 2011.
[19] DuyHoa Ngo, Zohra Bellasene, and Remi Coletta. A generic approach for combining linguistic and context profile metrics in ontology matching. In *ODBASE Conference*, 2011.
[20] Shvaiko Pavel and Jerome Euzenat. Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, 99, 2011.
[21] Giorgos Stoilos, Giorgos B. Stamou, and Stefanos D. Kollias. A string metric for ontology alignment. In *ISWC Conference*, pages 624–637, 2005.
[22] William Sunna and Isabel F. Cruz. Structure-based methods to enhance geospatial ontology alignment. In *GeoS*, pages 82–97. Springer, 2007.
[23] Amos Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.
[24] Peng Wang and Baowen Xu. Lily: Ontology alignment results for oaei 2009. In *OM*, 2009.