Opening the Black Box of Ontology Matching

DuyHoa Ngo, Zohra Bellahsene, Konstantin Todorov

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

Abstract. Due to the high heterogeneity of ontologies, a combination of many methods is necessary in order to discover correctly the semantic correspondences between their elements. An ontology matching tool can be seen as a collection of several matching components, each implementing a specific method dealing with a specific heterogeneity type (terminological, structural or semantic). In addition, a mapping selection module is introduced to filter out the most likely mapping candidates. This paper proposes an empirical study of the interaction between these components working together inside an ontology matching system. By the help of datasets from the Ontology Alignment Evaluation Initiative, we have carried out several experimental studies. In the first place, we have been interested in the impact of the mapping selection module on the performance of terminological and structural matchers revealing the advantage of using global methods vs. local ones. Further, we have carried an extensive study on the flaw of the performance of a structural matcher in the presence of noisy input coming from a terminological method. Finally, we have analyzed the behavior of a structural and a semantic component with respect to inputs taken from different terminological matchers.

1 Introduction

The field of ontology matching has matured considerably as a result of more than a decade of research and practice. Many ontology matching approaches and systems have been developed dealing with the semantic heterogeneity problem by taking into account various aspects of this problem [20]. Methodologically speaking, these approaches rely on techniques from fields as diverse as machine learning, graph matching, information retrieval, relational algebra, logics, – each of these fields providing a framework to deal with a certain heterogeneity type. In this respect, a standalone ontology matching system is a successful combination of several matching components. We consider that time has come to pay attention to the way these components connect to each other within a matching system and how these interactions impact the overall quality of the produced alignments.

Many challenges stand in front of the ontology matching community – a full picture is given in [20]. By this study, we contribute to the solution of matcher selection and combination problems, which are fundamental for the development of a stable system. A matching system can be seen as a combination of four main components: a terminological, a structure-based and a semantics-based matcher accompanied by a mapping selection module[1]. Although these components exploit different features of the entities

[1] Not each of these components is always and necessarily part of the system's architecture.

of an ontology, they are not independent. A structure-based matcher takes as an input the mappings resulting from a terminological matcher [1, 7, 24]; a semantics-based matcher may take as an input the mappings resulting from either a terminological [5, 9], or a structure-based matcher, or a combination of the mappings resulting from both [4, 6]. Therefore, difficulties can arise not only inside each component but also on the interaction lines between them. We take into consideration several of these difficulties.

A ***mapping selection*** module is usually introduced in order to filter out the best mapping candidates, at each of the different matching levels. The interaction of this module with the matchers is, therefore, among the basic issues to be addressed.

A ***terminological matcher*** discovers mappings by comparing annotations (i.e., labels, comments) of entities. To this end, it may use many different similarity measures. The difficulty is, on the one hand how to select the most appropriate similarity measures and, on the other hand, how to effectively combine them.

A ***structure-based matcher*** discovers mappings between entities by analyzing the similarity of the structural patterns, which these entities are part of. However, according to [3], almost all methods of this type are not stable and do not improve the matching quality when the structures of the ontologies are different. Moreover, structural matchers are error-prone, since they strongly depend on initial mappings provided by a terminological matcher and on the specific settings of the mapping selection component.

A ***semantic matcher*** is mainly used to refine candidate mappings [5, 6, 9]. It exploits the semantic constraints between entities in the ontologies in order to discover conflicts between potential mappings and remove them from the list of candidate mappings. To do that, in some tools [5, 9], the semantic module requires a confidence value for each mapping candidate. Then, it applies a global optimization method in order to find the minimal inconsistent set of mappings. Therefore, similarly to structural methods, semantic methods are error-prone because they also depend on the confidence values of the mappings obtained at previous steps.

This empirical study aims to investigate the interconnections between the different components in an ontology matching system. Our intention has been to make explicit the relations between these components by showing how one impacts the other and thus guide practitioners and researchers in the choice of the matchers with regard to the global quality of the matching system.

The rest of the paper is organized in the following manner. In the next section, we present a generic ontology matching system architecture together with an evaluation scenario for the ontology matching task. We continue by presenting two basic studies. With respect to different settings of the mapping selection module, we evaluate the performance of different terminological methods (Sections 3) and different structural methods (Section 4). Further, we go into a detailed study of the interaction between terminological, structural and semantic methods. We first study the performance of different structural matchers at the presence of noisy input (Section 5) and then the behavior of structural and semantic matchers with respect to mappings produced by different terminological methods that they take as an input (Section 6). Sections 3 to 6 present one independent study each and are structured in an uniform manner: the matching methods are presented first, followed by a presentation of the evaluation data and strategy and, finally, the results are given and supported by an in-depth discussion.

2 A Generic Framework for Ontology Matching and Evaluation

The main components of a standalone ontology matching system are depicted in the lower part of Fig. 1. As discussed in the introduction, the three core matching components are based on terminology, structure and semantics. The role of these matchers is to discover correct mappings or remove incorrect ones according to specific features extracted from the entities of the input ontologies. Additionally, a mapping selection module is introduced to act as a filter which selects the best candidate mappings. We define a matching strategy as the way the matcher and selection components work together in order to produce an alignment.

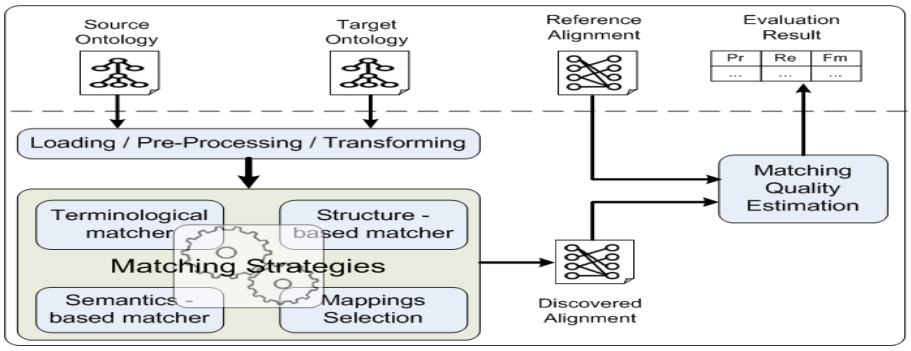

Fig. 1: Ontology Matching: System Architecture and Evaluation Scenario

To perform an evaluation of the quality of the different matching strategies, the ontology matching system requires matching scenarios as an input (upper part of Fig. 1). A matching scenario consists of a source and a target ontology and a reference alignment provided by a domain expert. Given a matching scenario, input ontologies are loaded, pre-processed and transformed into internal data structures (Loading/Pre-Processing/Transforming component). The Matching Quality Estimation module evaluates the quality of a given matching strategy by comparing discovered alignments with the reference alignment. It outputs three evaluation values corresponding to Precision (Pr), Recall (Re) and F-measure (Fm). In this study, we compute the harmonic means of precision, recall and F-measures on a set of n tests. These evaluation measures are used in the OAEI campaign and are given as follows.

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

For the *i*th test, $|A_i|$ denotes the total number of mappings discovered by a matching system, $|C_i|$ – the number of correct mappings, and $|R_i|$ – the number of reference mappings provided by a domain expert. In the sequel, all results will be given by considering F-measures only.

By following this generic architecture, we have developed the YAM++ system[2]. Various matching methods inside of the three matcher components and several filtering

[2] YAM++ - (not) Yet Another Matcher, published here: http://www2.lirmm.fr/ dngo/.

methods used in the mapping selection module have been implemented. The system is described in detail in [17]. Because of the broad scope and diversity of the techniques employed by YAM++, as well as its excellent results in the OAEI campaigns[3], we have used this system in order to evaluate the different matching strategies based on the interaction of the matchers. More detail about the setup of the matching and filtering methods for each matching strategy will be given in each experiment in the succeeding sections.

Note that required computation times for each technique have not been taken into account in this study. They could, however, be a useful decision factor when two techniques produce similar results in terms of precision and recall.

3 Terminological Matchers and Mapping Selection

In this evaluation, we focus on terminological matchers and we study their interaction with the mapping selection module. According to a classification found in [3], the terminological matching approaches are divided into ***local*** and ***global*** methods. Local methods focus on the similarity between individual entities, whereas global methods combine local ones, taking into account the semantic context that these entities belong to. We are particularly interested in the comparison between local and global methods.

3.1 Methods

We have considered several state-of-the-art local methods as well as advanced global methods, some of which have been proposed originally for YAM++.

Local Terminology-Based Methods We have implemented more than fifty local methods used for terminology-based matching [16]. We divide them into three groups based on the algorithm for computing similarity between strings that they rely on. To economize space, the following representative methods will be used in this experiment:

Edit distance-based methods. The similarity of two strings is computed based on the number of edit operations needed to transform one string to another. We have considered Levenstein and ISUB [21].

Token-based methods. These methods split strings into sets of tokens and then compare tokens by string-based methods. We have considered QGrams and TokLev (using Levestein to compare tokens).

Hybrid methods. Methods in this group split strings into sets of tokens and then compare tokens by combining string-based and linguistic-based methods. We have taken as examples HybLinISUB and HybJCLev. HybLinISUB uses a combination of ISUB and Lin [11]; HybJCLev relies on the Levenstein and the Jiang-Corath [8] methods.

Global Terminology-Based Methods In our experiments, we have implemented the following global methods:

[3] In OAEI 2012 YAM++ was first on the Conference, Multifarm, Benchmark and Bio-Medical track, and second on the Anatomy track.

Weighted Average with Local Confidence (LC). Each local method is assigned a local confidence value. These values are used as weights in a weighted average function to compute the final similarity score between entities. More details can be found in [1].

Harmony-based Adaptive Similarity Aggregation (HADAPT). Here, each local method is assigned a weight which is computed by the harmony estimation algorithm [12]. Then, a weighted sum aggregation method is used to produce a final similarity score between entities.

Machine Learning-Based Approach (ML). This method combines all local methods and constructs a classification function on the basis of given training data. In a machine learning setting, the training dataset consists of pairs of entities (seen as training examples) for which the confidence value of their similarity is known. Based on these training examples, a classifier learns a function which is able to predict the confidence value of an unseen pair of entities. In that way, the ontology matching task is transformed into a classification task. After testing the performance of over 15 machine learning techniques, we have seen that J48 decision tree is the most appropriate one for the ontology matching task [17]. This is the method that has been used in the following experiments.

Information Retrieval-Based Approach (IR). This method judges the similarity between two entities by the amount of overlap of the information content of their labels [18]. It splits all labels of entities into tokens and calculates the information content of each token in the whole ontology. Then, IR extends Tversky's similarity measure [23] with weight of tokens to compute a similarity score between labels of entities. The method compares similarity of two labels by using not only the sequence of characters themselves, but also their information content in an ontology. We will illustrate this idea by examples in this experiment.

3.2 Matching and Evaluation Strategy

To perform this experiment, we have chosen the Conference dataset from the OAEI including 21 test cases[4]. The reason for this choice is that this dataset consists of moderate-sized real-world ontologies describing the same domain. These ontologies are highly heterogeneous since they were developed by different people, hence, the same concept is often labeled differently. We assume that high matching quality of a system on these tests guarantees similar results of the system when applied to other real matching scenarios.

The matching evaluation strategy works as follows. For each matching method (including local and global ones) in the terminological matching module, we compute a similarity score for all pairs of entities of the input ontologies. The mapping selection module then selects candidate mappings according to the filter threshold value. At each level of this threshold, a H-mean F-measure is computed over all test cases in the dataset.

[4] http://oaei.ontologymatching.org/2012/conference/index.html

3.3 Results and Discussion

Fig. 2 shows the results of this comparison. As can be seen, almost all local methods (except for QGrams and ISUB) improve the F-measure when the threshold value of the filter increases. When the filter threshold increases, it appears that only highly similar or identical labels are be passed. Therefore, as Fig. 2 shows, local methods closely converge to results of the Identical method (≈ 0.55) when the filter threshold reaches 0.95 and 0.97. The same trend is observed for HADAPT and LC because they are linear combinations of local methods.

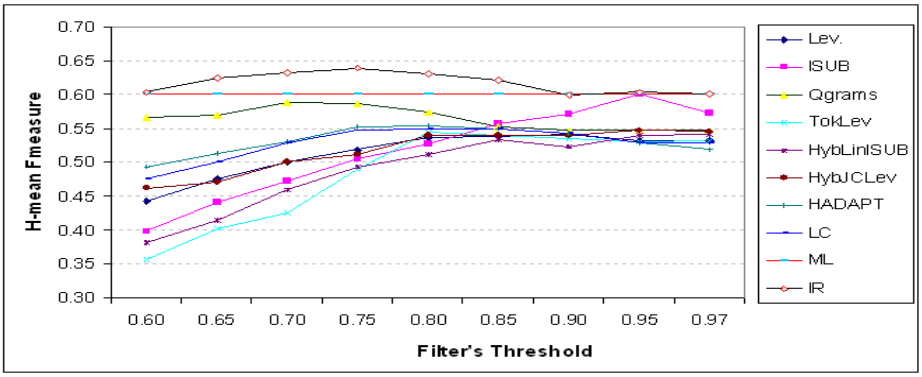

Fig. 2: Mapping Selection for the Terminological Matcher Module

The experiment shows that the two global methods, ML and IR outperform the other techniques within the terminological matcher module. Therefore, in what follows, we will discuss in more detail these two methods.

Performance of ML A machine learning method requires training data on which to learn a classification function and test data on which to apply this function. To create training data independent on the Conference dataset on which the evaluation will be performed, we have used data from the OAEI Benchmark 2009[5] and I3CON[6] datasets. We have constructed 10 different training datasets by using these two sources and we have trained the decision tree ML method on each of these 10 training sets. At each time, the learned classification algorithm has been applied to the Conference dataset, providing 10 different results. The result given in Fig. 2 is obtained by taking the average over these 10 results.

We note that the ML method does not depend on the filter threshold since no candidate mapping selection takes place. As it is seen from the figure, ML returns a better matching quality than LC, HADAPT and all local methods. For example, the ML method discovers (cmt.owl#Co-author $\equiv$ conference.owl#Contribution_co-author) in the `cmt.owl` and `conference.owl` ontologies, whereas local methods return a low similarity score between these labels (Levenstein(Co-author,Contribution_co-author) = 0.4; QGrams(Co-author,Contribution_co-author) = 0.6). This is explained by the fact

[5] http://oaei.ontologymatching.org/2009/benchmarks
[6] http://www.atl.external.lmco.com/projects/ontology/i3con.html

that ML does not use arithmetic combination functions like LC and HADAPT, instead, it extracts the combination rules on local methods from training data. ML is able to find many patterns in the training data similar to the current example (e.g., (networkA.rdf#Office ≡ networkB#OfficeSoftware), (russia1#payment ≡ russia2#means_of_payment), etc.). However, the 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 data, ML can discover (cmt.owl#Co-author ≡ conference.owl#Contribution_co-author), but not with other. Moreover, for a given training data this mapping is discovered by ML, but (cmt.owl#Document ≡ conference.owl#Conference_document) is not, even though the latter seems similar to the former. To address this problem, we have designed the IR method, which is discussed in the sequel.

Performance of IR The IR method proposed in YAM++ [18] outperforms all other methods in the experiment. We analyze this fact by giving an example with two entities: `cmt.owl`#Co-author and `conference.owl`#Contribution_co-author. After splitting and normalizing the labels, we have the following two sets of tokens: {`coauthor`} and {`coauthor`, `contribution`}. Token `coauthor` appears in each input ontology only once, whereas token `contribution` appears 10 times among 60 concepts in the `conference.owl` ontology. Therefore, the information content of the token `contribution` is lower than that of token `coauthor`. In particular, the normalized *tf-idf* weights of each token inside the input ontologies are equal: $\{w_{coauthor} = 1.0\}$, $\{w_{coauthor} = 1.0, w_{contribution} = 0.34\}$. The two sets of tokens share only the token `coauthor`, hence the similarity computed by Tversky's method is $\frac{1.0+1.0}{1.0+1.0+0.34} = 0.855$. Similarly, we have the similarity between (`Document`,`Conference_document`) equaling `0.91`. In this pair, the token `conference` appears 15 times in the `conference.owl` ontology. Therefore, this token brings little information for this ontology and, consequently, this pair of entities represents a likely match.

It is difficult to give a clear indication which of these two best performing methods to use – ML or IR. Clearly, in the absence of training data, the choice will go for IR. Even in the presence of training data, the IR method appears to be more suitable because it reaches an overall higher performance than ML for low values of the mapping selection threshold. However, an advantage of the ML method is that it does not depend on the setting of a filter threshold. Both methods can be combined within the architecture of an ontology matching tool.

4 Structural Matchers and Mapping Selection

In this evaluation, we are interested in the behavior of structural similarity methods with respect to the mapping selection module.

4.1 Methods

The following standard structural matching methods have been considered within this study: ***ANCESTORS*** (two entities are similar if all or most of their ancestor entities are already similar), ***DESCENDANTS*** (two entities are similar if all or most of their descendant entities are already similar), ***LEAVES*** (two entities are similar if all or most

of their leaf entities are already similar [2]), ***ADJACENTS*** (two entities are similar if all or most of their adjacent entities (parents, children, siblings, domains, ranges) are already similar), ***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 [10]). ***DSIPATH (Descendant's Similarity Inheritance)*** (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]), and ***SSC (Sibling's Similarity Contribution)*** (two entities are similar if the total contribution of their sibling entities is higher than a specific threshold [22]).

Additionally, we have considered the ***SP (Similarity Propagation)*** method. This method is proposed in the system YAM++ as an extension of the well-known similarity flooding algorithm [15]. The basic idea of the method is as follows. Assume that the entities A_1 and A_2 in one ontology are related by a directed relation P and the entities B_1 and B_2 in another ontology are related by the same directed relation. Then, if we discover that (A_1, B_1) is a match, the SP method would imply that (A_2, B_2) is a match, too. The similarity values between the two pairs are propagated to each other at each iteration of algorithm. The approach is described in detail in [19].

4.2 Matching and Evaluation Strategy

To perform this experiment, we have used the Benchmark 2011 dataset from the OAEI campaign including 103 test cases. These datasets are acquired by taking an original ontology and altering the names of some of its entities by using random strings (no variation by naming convention or synonym words). The entities whose labels have not been altered are kept as in the original ontology. Therefore, a matching scenario which takes as an input the original and the altered ontologies is appropriate for evaluating the performance of structural methods. As an input to these structural methods, we use the alignment produced by an identical metric (defined as one which returns a correct mapping only for identical strings), since the non-altered string names are identical in both ontologies. An additional characteristics of this dataset is that in some tests, not only the names of entities are altered but also the ontology structure by flattening, extension and other structure modifying operations.

The matching and evaluation strategy used in the experiment is given as follows. Only three modules will be used: a terminological matcher, a structure-based matcher and a mapping selection module.

The ***terminology-based*** matcher provides input mappings to the structural matcher. They are provided by the *identical metric*, denoted by INIT_MAPPINGS.

Each ***structure-based*** matcher corresponding to each of the selected structural measures above produces a similarity matrix for all pairs of entities from the two input ontologies.

In the ***mapping selection*** module, we vary the threshold (0.01 – 0.9) to filter out the mappings discovered by this matcher. The mappings obtained by the structural matcher are combined with mappings obtained by the terminological matcher to produce the set of candidate mappings. Then, a greedy selection method [14] is used to extract the final alignment.

4.3 Results and Discussion

As can be seen from Fig. 3, when the threshold varies from 0.6 to 0.9, the structural methods converge to the INIT-MAPPINGS line where H-mean Fmeasure = 0.68 (4463 correct mappings, 27 incorrect mappings, 4342 unfound). This means that the structural methods did not discover additional correct mappings or they discovered correct mappings, which already exist in the init mappings. This is natural, because most of the structural methods compute similarity between two entities based on the overlap of their structural patterns (i.e., adjacent, ancestor, etc.) by using, for instance, the Jaccard measure. Therefore, the higher the filter threshold, the lower the possibility of discovering new mappings.

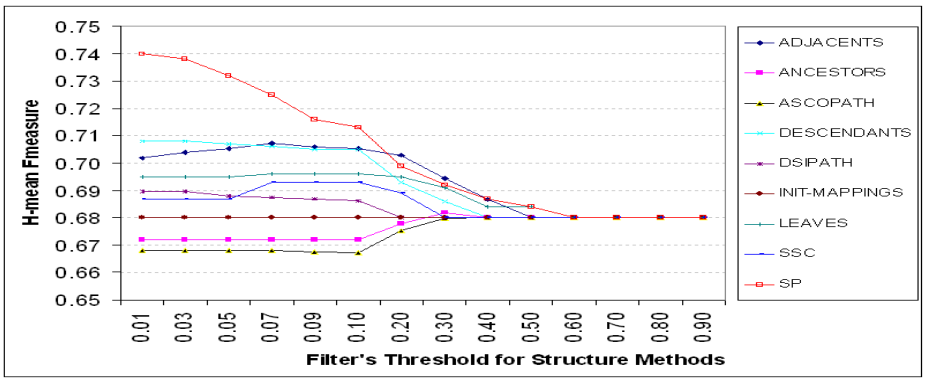

Fig. 3: Mapping Selection for Structural Methods

We note that the corresponding matching qualities of the structural methods differ significantly when the filter threshold is set to small values. We will consider methods that perform poorly and such that perform well.

For threshold values between 0.01 and 0.09, ASCOPATH and ANCESTORS discover many incorrect mappings. For example, when the threshold is equal to 0.01, ACSOPATH discovers 90 $(= 4733 - 4643)$ additional correct mappings but 453 $(= 480 - 27)$ incorrect mappings in comparison to the init mappings. This can be explained as follows. After observing the ontologies in the Benchmark 2011 dataset, we see that the maximum depth and also maximum number of ancestors of an entity in the ontology hierarchy is 5. Assume that two entities have only one common entity among their ancestors. Then their similarity score is equal to $1/10 = 0.1$ at least. If two entities do not have any common entities, then their similarity is equal to 0. Therefore, with a threshold in the range of 0.01 to 0.09, any pair of entities having at least one common ancestor will be considered as a match. Since sibling entities have the same ancestors and paths to these ancestors, they will have the same structural patterns. Therefore, many pairs of entities will have the same similarity scores. Moreover, one entity may have many descendant entities so many pairs of entities can be coupled, consequently, many incorrect mappings will be produced.

In contrast, other methods such as DESCENDANTS, LEAVES, DSIPATH and SSC provide better results with small thresholds than the methods discussed above. They dis-

cover more additional correct mappings and, consequently, improve the overall quality of the matching. For example, with a threshold equal to 0.01, DESCENDANTS discovers 494 $(= 5137 - 4643)$ additional correct mappings and 175 $(= 202 - 27)$ incorrect mappings in comparison to the init mappings. Similarly to the 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 one entity is important to another [22]. Therefore, by running greedy selection, which always selects the pair of entities having high similarity score with 1:1 cardinality, most of the selected mappings are correct.

Performance of SP The similarity propagation (SP) method that we propose differs from the other structural methods discussed above in several aspects. Note that the similarity scores produced by SP are not absolute but relative values due to the normalization process at the end of each running iteration. SP propagates similarity values from one pair of entities to another, hence, if two entities have a similarity score higher than 0, then they are considered as similar to a certain degree. Thus, with a low threshold filter, SP discovers more correct mappings than with a high threshold value. Moreover, the similarity score of a pair of entities depends not only on their current status but also on the status of other related (neighboring) pairs. The more neighbors with high similarity a pair of entities has, the likelier that they are matched. For example, when the threshold is set to 0.01, SP discovers 1298 $(= 5941 - 4643)$ additional correct mappings and 247 $(= 274 - 27)$ incorrect ones in comparison with the init mappings. Therefore, SP distinguishes well correct and incorrect mappings by ranking the similarity scores which is the main reason why this method outperforms the other local structural methods discussed above when the filter threshold is low.

5 Impact of Noisy Input on Structural Matchers

In this experiment, we evaluate the behavior of different structural matchers when we add noise into the mappings that these methods take as an input from a terminological matcher. Here, we call "noise" a pair of dissimilar entities labeled as similar by the terminological matcher. Indeed, in real matching scenarios, a terminological method rarely produces 100% precision, consequently, it rarely provides input mappings without noise to the structural methods. We will study the impact of this noise on the mappings discovered by several structural methods with the aim to outline the most stable among these methods with respect to the presence of noise.

5.1 Methods and Evaluation Strategy

For these experiments, we have used the Benchmark 2011 dataset from the OAEI campaign. The reasons for this choice given in Section 4 are valid here, as well, since we are dealing with structural methods.

At ***terminological level***, we use the identical metric. To produce noise, we add a number of random incorrect mappings, which correspond to $N\%$ of the size of the original init mappings, with $N \in \{0, 10, .., 100\}$.

At ***structural level***, the matcher takes input mappings from the terminological matcher. According to the experiments in Section 4, we select the best threshold filter for each structural method. For example, $\theta_{SP} = 0.01$, $\theta_{DESCENDANTS} = 0.01$, $\theta_{ADJACENTS} = 0.07$, etc.

At each iteration, we count the total number of correct mappings and the total number of incorrect mappings that a structure method produces over all 103 test cases contained in the Benchmark 2011 dataset.

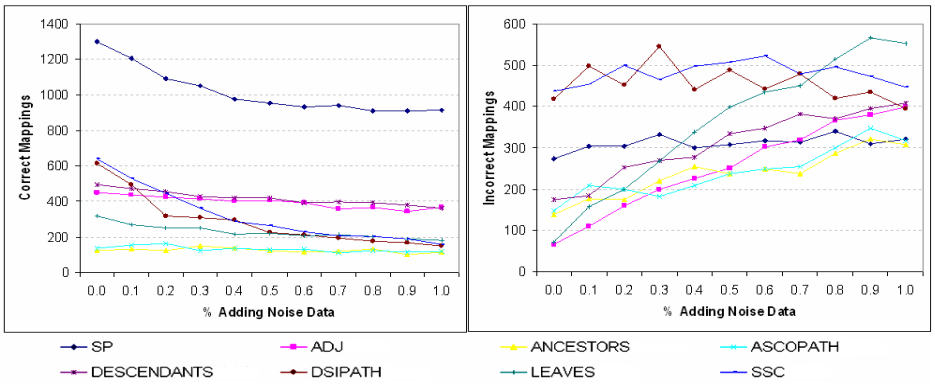

Fig. 4: Impact of input noise on structural matchers.

5.2 Results and Discussion

Fig. 4 shows the total number of correct and incorrect mappings produced by the structural methods at each time when noisy data are added to the input. When noisy data are added, the number of correct mappings discovered by all the methods decreases. Regarding the number of incorrect mappings, it increases for all methods, except for DSIPATH and SSC. Note that DSIPATH and SSC differ from the other local structural methods in terms of the interaction between the entities in an ontology. For example, the similarity of two entities computed by DSIPATH strongly depends on the similarity provided by input mappings and decreasingly depends on the similarity of their parents, grandparents, etc. Consider two entities of two input ontologies. If noise appears at the same level in their paths to the root, their similarity will be impacted by this noise, otherwise, it will not. Therefore, the impact of noise in discovering further mappings depends on the position of the entities in the hierarchies of the input ontologies. Because noise is generated randomly, its impact is hard to predict for these methods. Other structural methods use set operations (i.e., intersection, union) with no hierarchical consideration for the elements. When noise appears in the set of ancestors or descendants of two entities, the noise will directly propagate errors to them. Therefore, as seen in Fig. 4, the number of incorrect mappings increases in almost all structural methods of this type.

This experiment shows the dominance of similarity propagation (SP) over other structural methods in terms of stability. When noisy data reaches 100%, SP still discovers 913 additional correct mappings in comparison to the init mappings. Note that the maximum number of correct mappings discovered by the other methods is only 612 mappings with no noise added. Moreover, from 0% to 100% of the noisy data, SP

produces only 57 $(321 - 274)$ additional incorrect mappings. In contrast, for example, the LEAVES method produces 481 $(553 - 72)$. This is explained by the fact that SP takes into account all kinds of semantic relations of entities such as concept-concept, concept-property and property-property, which reduces the impact of noise.

6 Interaction of Terminological Matchers with Structural and Semantic Matchers

In this evaluation, we are going to study the impact of the quality of the input mappings provided by several terminological methods on the matching quality of structural and semantic matchers. More precisely, we are interested in discovering which are the terminological methods that provide best performance of the structural and semantic matchers for a given mapping selection threshold.

6.1 Methods and Evaluation Strategy

To carry out this experiment, we have used the Conference dataset from the OAEI campaign, which is a real world dataset from the domain of scientific publishing. Our evaluation strategy is described as follows.

At ***terminological level***, we have used three different methods to produce initial mappings. The choice of these matchers has been motivated by the study described in Section 3 and the results shown in Fig. 2. We have chosen QGrams representing token-based methods and ISUB representing edit-based methods because they show different behaviors when the terminology-based filter threshold changes as compared to the other methods. In addition, we have included IR, representing global methods, which is the best performing among these methods.

At ***structural level***, we have considered SP which takes input from the terminological matchers and performs similarity propagation. This choice is justified by the fact that this method has shown to perform best in the experiments in Section 4.

At ***semantic level***, we use the global diagnosis optimization method proposed in [13] which refines input terminological mappings in order to remove inconsistent ones.

We have studied the performance of each of the terminological methods when used alone and when used as an input for the structural and the semantic methods, respectively. At each iteration, the matching quality is evaluated by comparing the discovered alignment to a reference alignment.

6.2 Results and Discussion

Fig. 5 shows the performance of the terminological methods used alone and in combination with a structural matcher (SP) again as a function of the mapping selection threshold. Fig. 6 shows the behavior of the same terminological methods, this time taken as an input by a semantic matcher. The first observation is that the structural and the semantic methods combined with terminological matchers have similar behaviors, therefore the following analysis will encompass both.

Globally, the combined methods outperform the single terminological methods. Similarly to the previous experiments, the overall performance increases by increasing the threshold value. Quite straightforwardly, the quality of the combined methods increases simultaneously with the quality of the single terminological methods.

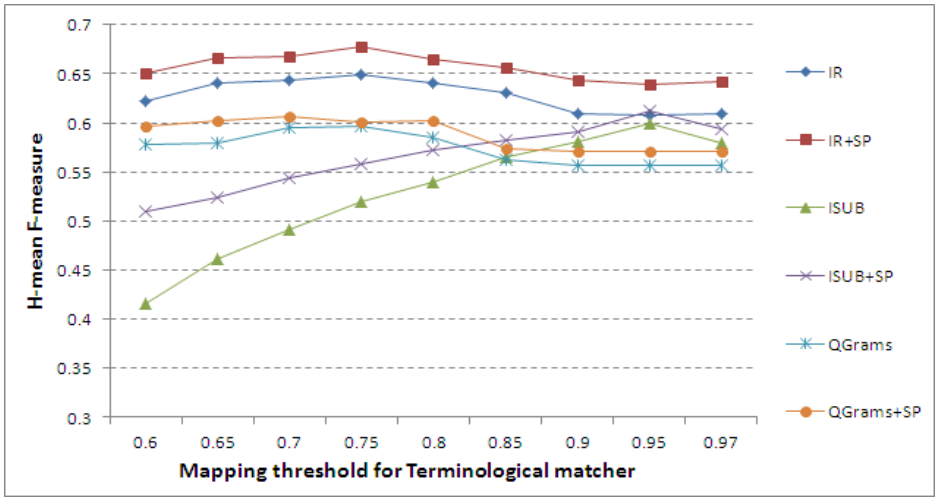

Fig. 5: Interaction of terminological methods with a structural matcher (SP) w.r.t. different values of the mapping selection filters.

Further, we notice that the methods based on QGrams tend to be more stable over the variations of the filter threshold and provide high quality results already at low filter values. This is explained by the fact that the QGrams measure is based on Jaccard similarity computation and as soon as the threshold value reaches 0.6, the matcher already accounts for two third of the overlapping tokens. The methods based on ISUB have a different behavior – they have an almost linear growth of the performance as a function of the filter threshold, reaching higher values of the F-measure than the ones of the QGrams methods for thresholds above 0.9, both for structural and semantic approaches.

We explain that by the fact that at a certain level of the threshold value, the number of incorrect mappings is always higher than the number of correct mappings, especially due to the 1:1 cardinality. Therefore, when the threshold value increases, the number of removed incorrect mappings will get higher than the number of removed correct mappings. Thus, the overall quality increases. However, after surpassing the threshold of 0.95 the quality decreases again. This is due to the fact that when the threshold is that high, only identical or nearly identical strings are passed (i.e. the overall number of passed entities decreases).

Finally, we note that the mapping selection component is a very important intermediate level between the terminology matchers and the structural or semantic ones in order to select output of each matching component. Indeed, the quality of the produced alignments is much worse if no mapping selection is performed. In the experiments, the role of mapping selection is shown by varying the value of the filter threshold.

As a general conclusion, we outline the fact that both the structural and the semantic matchers boost the performance of both local and global terminological methods, but perform best by taking input from the global IR method.

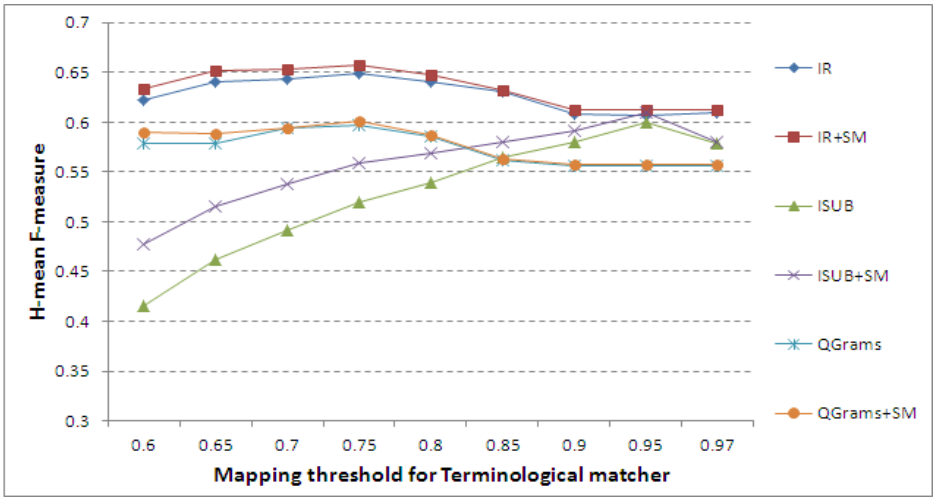

Fig. 6: Interaction of terminological methods with a semantic matcher (SM) w.r.t. different values of the mapping selection filters.

7 Conclusion

In this empirical study, we have presented an analysis of the interaction between the components of an ontology matching system, seen as a chain in which the resulting mapping of a given module is the input to another. We have used evaluation data from the OAEI campaign. In the first place, we were interested in the impact of the mapping selection module on the performance of terminological and structural methods revealing the advantage of using global methods vs. local ones. Further, we have carried an extensive study on the flaw of the performance of a structural method in the presence of noisy input coming from a terminological method. Finally, we have analyzed the behavior of a structural and a semantic matcher with respect to different inputs taken from different terminological methods at different values of the mapping selection filter.

The results of this study are oriented towards researchers and practitioners and are meant to serve as a guide in the design and the use of a matching tool. Our outcomes provide a support on the choice of matchers and the effects that can be expected in their selection and combination. The ultimate goal has been to give the user control and understanding of the mechanism behind an ontology matching system.

References

[1] I. F. Cruz, C. Stroe, M. Caci, F. Caimi, M. Palmonari, F. P. Antonelli, and U. C. Keles. Using agreementmaker to align ontologies for oaei 2010. In *OM*, 2010.
[2] R. Dieng and S. Hug. Comparison of personal ontologies represented through conceptual graphs. In *ECAI*, pages 341–345, 1998.
[3] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg, 2007.
[4] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match: an algorithm and an implementation of semantic matching. In C. Bussler, J. Davies, D. Fensel, and R. Studer, editors, *The Semantic Web: Research and Applications*, volume 3053 of *LNSC*, pages 61–75. Springer Berlin Heidelberg, 2004.

[5] J. Huber, T. Sztyler, J. Nößner, and C. Meilicke. Codi: Combinatorial optimization for data integration: results for oaei 2011. In *OM*, 2011.
[6] Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka. Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):235–251, 2009.
[7] N. Jian, W. Hu, G. Cheng, and Y. Qu. Falcon-ao: Aligning ontologies with falcon. In *Proceedings of K-CAP Workshop on Integrating Ontologies*, pages 85–91, 2005.
[8] J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, 1997.
[9] E. Jiménez-Ruiz and B. Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, and E. Blomqvist, editors, *The Semantic Web ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 273–288. Springer Berlin Heidelberg, 2011.
[10] B. Thanh Le, R. Dieng-Kuntz, and F. Gandon. On ontology matching problems. In *ICEIS (4)*, pages 236–243, 2004.
[11] D. Lin. An information-theoretic definition of similarity. In *ICML*, pages 296–304, 1998.
[12] M. Mao, Y. Peng, and M. Spring. A harmony based adaptive ontology mapping approach. In *SWWS*, pages 336–342, 2008.
[13] C. Meilicke. Alignment incoherence in ontology matching. In *Thesis*, 2011.
[14] C. Meilicke and H. Stuckenschmidt. Analyzing mapping extraction approaches. In *OM*, 2007.
[15] 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.
[16] D. Ngo, Z. Bellahsene, and R. Coletta. A generic approach for combining linguistic and context profile metrics in ontology matching. In R. Meersman, T. Dillon, P. Herrero, A. Kumar, M. Reichert, L. Qing, B.-C. Ooi, E. Damiani, D.. Schmidt, J. White, M. Hauswirth, P. Hitzler, and M. Mohania, editors, *On the Move to Meaningful Internet Systems: OTM 2011*, volume 7045 of *Lecture Notes in Computer Science*, pages 800–807. Springer Berlin Heidelberg, 2011.
[17] D. Ngo, Z. Bellahsene, and R. Coletta. A flexible system for ontology matching. In S. Nurcan, editor, *IS Olympics: Information Systems in a Diverse World*, volume 107 of *LNBIP*, pages 79–94. Springer Berlin Heidelberg, 2012.
[18] D.H. Ngo. *Enhancing Ontology Matching by Using Machine Learning, Graph Matching and Information Retrieval Techniques*. PhD thesis, Univrsity of Montpellier 2, 2012 (In print).
[19] D.H. Ngo, Z. Bellasene, and R. Coletta. Yam++ results for oaei 2011. In *OM*, 2011.
[20] P. Shvaiko and J. Euzenat. Ontology matching: State of the art and future challenges. *IEEE TKDE*, 99, 2011.
[21] G. Stoilos, G. Stamou, and S. Kollias. A string metric for ontology alignment. In Y. Gil, E. Motta, V.R. Benjamins, and M.. Musen, editors, *The Semantic Web ISWC 2005*, volume 3729 of *LNCS*, pages 624–637. Springer Berlin Heidelberg, 2005.
[22] W. Sunna and I.. Cruz. Structure-based methods to enhance geospatial ontology alignment. In F. Fonseca, M.A. Rodriguez, and S. Levashkin, editors, *GeoSpatial Semantics*, volume 4853 of *Lecture Notes in Computer Science*, pages 82–97. Springer Berlin Heidelberg, 2007.
[23] A. Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.
[24] P. Wang and B. Xu. Lily: Ontology alignment results for oaei 2009. In *OM*, 2009.