ABSTRACT: Ontology matching is one of the most important works to achieve the goal of the semantic web. To fulfill this task, element-level matching is an indispensable step to obtain the fundamental alignment. In element-level matching process, previous work generally utilizes multiple measures to compute the similarities among elements, and then combine these similarities using a weighted sum formula to determine later the semantic correspondences. The proper selection and combination of these similarities strongly influence the final quality of the matching system.

In this paper, we introduce element-level ontology matcher as machine learning system which utilizes regression algorithms to automatically find many possible combination relations of similarity measures. We adopt different similarity measures to extract learning features, and train the system on sample data from Benchmark and Conference tracks OAEI 2015. To match two ontologies, after training, we measure different similarities for each entity pair, and predict the overall similarity of each pair using the learned regression model to get similarity matrix. After that we extract correspondences by applying naïve descending algorithm on the similarity matrix. These correspondences are filtered by previously known semantic techniques to enhance matching precision while preserving matching recall.

Experimental results, using dataset in conference track from OAEI 2015, show that our extracted similarity features are efficient in terms of f-measure evaluation criteria, and outperform the widely used measures in ontology matching systems. Moreover, our method for similarity combination which depends on regression model outperforms the present combination methods. Besides, in comparison to the matching systems participated in OAEI 2015 campaign, our matching system showed to be highly competitive and had a high ranking position.

Subject Categories and Descriptors: D.2.12 [Interoperability]: Data mapping; I.2.6 [Learning]: Parameter learning; I.2.7 [Natural Language Processing]: Language parsing and understanding.

General Terms: Terminological Heterogeneity, Similarity Measure

Keywords: Ontology Matching, REPTree, Semantic Similarity, Similarity Combination, Improve Recall

Received: 22 September 2017, Revised 2 November 2017, Accepted 9 November 2017

1. Introduction

The semantic web is receiving increasing attentions because of its bright future. In the semantic web, ontologies are essential components which are explicit specifications of conceptualization [1]. Thus, a large number of ontologies have been created in the last decade. Although many of them describe the same domain, they cannot share information with each other since they are designed by different conventions. Hence, ontology matching is required due to the heterogeneous and distributed nature of ontologies. Many ontology matching systems have been designed in recent years. The state of the art approaches have made a significant progress in ontology matching field, but none of them gain a clear success in terms of matching quality performance over all matching scenarios [3], especially in terms of recall. We testify that in the results of OAEI 2014 camping, where most systems have low recall with regard to precision. Ele-
ment level techniques [3] are widely utilized. These techniques take advantage of lexical information as essential elements and use string-based or language-based similarity measures to decide the matched pairs. Ontology matching systems differ in how these measures are employed and combined. Most of matching systems use some similarities and combine them using a weighted sum formula with different methods for determining the weights. Merely depending on a weighted sum formula to combine similarities is impractical, since it is not necessarily suitable for all heterogeneities and matching scenarios. As a result some correspondences will not be found and thus system recall will be low.

We aim to discover more correspondences, i.e. increase matching system recall. To do so we have designed five similarity measures that are based on previously known ones, then we have built our regression model using these measures as features. This model addresses the problem of terminological heterogeneity in ontology matching task since it combines different similarity measures from different categories each dedicated to different type of terminological heterogeneity. Combining these measures using the proposed regression model gives the ability to raise the number of discovered correspondences (matched pairs) by producing higher and more objective similarity values than the traditional combination techniques.

Compared with other automatic combination methods, such as harmony-based method and local confidence, the learned regression model outperforms these methods in terms of quality, especially recall. Furthermore the proposed features outperform the similarity measures they use.

The rest of this paper is organized as follows. Section 2 reviews related work. In Section 3, we describe the setup of the paper. Section 4 presents the designed features. In section 5 we describe the training process. In section 6 we describe the alignment extraction process. Section 7 shows the experiments and the analysis. We conclude this paper in the final section.

2. Related Work

Ontology matching, finding semantic correspondences, is a crucial part to achieve the goal of the semantic web. So far, dozens of ontology matching systems have been created [1]. Lexical information, including names and labels describing entities are valuable to matching systems. Essential correspondences between two ontologies are found by element-level matchers. Mainly, element-level matchers compute multiple similarities for each entity pair, and then combine these similarities in some way to get one similarity value that is used to decide if an entity pair are matched or not. There are many techniques for measuring similarity between ontology entities, some of them are [2]: String-based measures such as Levenstein, Jaro, JaroWinkler, Stoilos...Etc. Language-based measures [3] such as Lin, WuPalmer, JiangCorath...Etc. Context-based measures that suppose similar entities are found in similar context [4], in this case, a virtual document or a context profile is built for each entity.

Measures from those types were widely adopted in deferent ways by many matching system such as RI MOM [5], ASMOV [6], PRIOR+ [7], YAM++ [8], AgreementMaker [9], AML [10], CroMatcher [11] to measure similarities between ontologies entities. Moreover, there are many methods to combine multiple measures and get one overall similarity to determine later if the two entities are matched or not. These methods are [2]: Max/Min, Average, SIGMOID, Weighted product, and weighted sum. All of these methods just depend on one relation to combine similarity measures, but there might be many others. Furthermore, the weights in some of these methods (i.e. SIGMOID, weighted sum, weighted product) need to be adjusted. To overcome the limitation of manually adjusting weights, methods were developed to automatically adjust them. These methods are: Harmonic Adaptive Weighted Sum [7], developed in PRIOR+ matching system and local Confidence Weighted Sum [12], developed in AgreementMaker matching system.

Those methods are good in terms of adaptation and precision but they still have low recall. So, we propose in this paper an efficient similarity combination method that uses machine learning technique called REPTree to well estimate the overall similarity of entity pair given their different similarities. The features of the learning system are well designed in that they utilize similarity measures from different types to deal with different heterogeneities. This method significantly improves matching recall, but it slightly affects the precision. To deal with that, we apply the semantic filtering technique developed in ASMOV ontology matching system [6] to remove inconsistency correspondences, thus increasing precision.

3. Problem Statement

Ontology is a formal specification of a shared conceptualization [13]. We describe the ontology as a 6-tuple:

\[ O = (C, P, H^C, H^P, A, I) \]

Where \( C \) and \( P \) are the sets of concepts and properties, respectively. \( H^C \) defines the hierarchical relationships \( H^C \subseteq C \times C \). \((c_i, c_j) \in H^C\) denotes that concept \( c_i \) is the subconcept of \( c_j \). Similarly, \( H^P \) defines the hierarchical relationships between each property and its subproperties, \( H^P \subseteq P \times P \). \( A \) is the set of axioms. \( I \) is the set of instances of concepts and properties.

We refer to entities in ontology as concepts and properties. Entities have several lexical descriptions, i.e., names, labels and comments. For instance, an entity’s name is “Journal”; its label is “Journal or magazine” and its comment is “A periodical publication collecting works from different authors”.

53
We also define:

- \( \text{Hier}(e) \) denotes a set of subconcepts of concept \( e \), or a set of subproperties of property \( e \):

Given concept \( c \in C \),

\[
\text{Hier}(c) = \{ c_j | j \in [1, N_{hc}], (c_j, c) \in H^C \}
\]

Given concept \( p \in P \),

\[
\text{Hier}(p) = \{ p_j | j \in [1, N_{hp}], (p_j, p) \in H^P \}
\]

- \( \text{Domain}(p) \) is a set of concepts that have the property \( p \):

\[
\text{Domain}(p) = \{ c | j \in [1, N_{dp}], \text{ concepts having the property } p \}
\]

- \( \text{Range}(p) \) is a set of concepts, whose instances can be the value of the property \( p \):

\[
\text{Range}(p) = \{ c_j | j \in [1, N_{rp}], \text{ concepts whose instances can be the values of property } p \}
\]

- \( \text{Rest}(c) \) is a set of properties and concepts in which each property is a property of concept \( c \), and each concept is used to describe concept \( c \):

\[
\text{Rest}(c) = \{ e_j | j \in [1, N_{re}], \text{ properties or concepts used to restrict concept } c, \text{ excluding its hierarchical relationships} \}
\]

The task of ontology matching is to find the alignment between entities in a source ontology \( O_1 \) and a target ontology \( O_2 \). We define an alignment as a set \( \{(e_1, e_2, R, n) | e_1 \in O_1, e_2 \in O_2 \} \). Every element in the set is called a correspondence. \( e_1 \) is an entity in \( O_1 \), and \( e_2 \) is an entity in \( O_2 \). \( R \) is the relation type (equivalence \( \equiv \) or subsumption \( \subseteq \)). \( n \) is the confidence of the correspondence. The matching system presented here offers an implement of an element-level matcher based on machine learning with semantic filtering. We declare two preconditions in advance. First, the hierarchy structure is not dealt independently. It is because an independent structure-level matcher is error-prone and strongly depends on the results of element-level matchers [14]. But hierarchal relations (which are represented by \( H^C \) and \( H^P \)) are utilized by our model in one feature when the terminological description of an entity is not expressive then we use these relations to describe it by its sub entities’ descriptions, see section 4.5.

Second, the resulted correspondences only have equivalence \( (\equiv) \) relations.

4. Features Extraction

To get stable and good quality alignment we design our features such that they complement each other. Five features have been extracted each capture different terminological heterogeneities. These features are: string-based feature for syntactic similarity, language-string-based feature for semantic similarity with detecting compound words, weighted-language-string-based feature for semantic similarity discarding meaningless words, abbreviation-based feature for detecting abbreviation and IR-based feature for context similarity. The general algorithm is as follows:

\[
\text{Algorithm 1 features extraction algorithm}
\]

\[
\text{simTable, Entity}_1, \text{ Entity}_2 = \emptyset
\]

\[
\text{Entity}_1 = C_1 \cup P_1
\]

\[
\text{Entity}_2 = C_2 \cup P_2
\]

\[
\text{for each } e_1 \in \text{Entity}_1 \text{ do}
\]

\[
\text{for each } e_2 \in \text{Entity}_2 \text{ do}
\]

\[
f_1 = \text{syntacticSim}(e_1, e_2)
\]

\[
f_2 = \text{semanticSim1}(e_1, e_2)
\]

\[
f_3 = \text{semanticSim2}(e_1, e_2)
\]

\[
f_4 = \text{abbreviationSim}(e_1, e_2)
\]

\[
f_5 = \text{IRSim}(e_1, e_2)
\]

\[
\text{simTable} = \text{simTable} + (e_1, e_2, f_1, f_2, f_3, f_4, f_5)
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

\[
\text{return simTable;}
\]

We explain in the following subsections each of these features.

4.1 Syntactic similarity

This feature concerned with the case where two entities have names, labels or comments with similar characters, i.e. they are syntactically similar. There are many string-based techniques that can be used here, such as [15]: Levenstein, Hamming, SmithWaterman, needlemanWunch, Gotoh, Jaro, JaroWinkler, Stoilos(ISUB), Etc. From those, we select Stoilos similarity measure [16]. So, syntactic similarity of two entities \( e_1 \) and \( e_2 \) is the max Stoilos similarity of their names, labels, and comments:

\[
\text{syntacticSim}(e_1, e_2) = \max\{\text{Stoilos(name}_1,\text{name}_2),\text{Stoilos(label}_1,\text{label}_2),\text{Stoilos(comment}_1,\text{comment}_2)\}
\]

4.2 Semantic Similarity (1)

This feature detects entities that have descriptions with similar meaning according to WordNet, even if there is a compound word of two or more tokens. Techniques used here are: Lin similarity measure [3], Stoilos similarity [16] and SoftJaccard similarity combination function [15]. So, semantic similarity of two entities \( e_1 \) and \( e_2 \) is the max SoftJaccard-Lin-Stoilos similarity of their names, labels, and comments:

\[
\text{semanticSim1}(e_1, e_2) = \max\{\text{SoftJaccardLinStoilos}(\text{name}_1,\text{name}_2),\text{SoftJaccardLinStoilos}(\text{label}_1,\text{label}_2),\text{SoftJaccardLinStoilos}(\text{comment}_1,\text{comment}_2)\}
\]

The detailed algorithm for computing SoftJaccard-Lin-
Stoilos similarity of two strings $S_1$ and $S_2$ is as follows:

Algorithm 2

|totalSim, sim, sharedTokens, unsharedTokens, sharedTokensSim = 0
|Tokens$_1$ = tokenize, removeStopWord, stem($S_1$)
|Tokens$_2$ = tokenize, removeStopWord, stem($S_2$)
|for each $t_1$ $\in$ Tokens$_1$, do
|   for each $t_2$ $\in$ Tokens$_2$, do
|     if $t_1$, $t_2$ in WordNet then
|       sim = Lin($t_1$, $t_2$)
|     else
|       sim = Stoilos($t_1$, $t_2$)
|     if Sim > threshold then
|       sharedTokensSim = sharedTokensSim + sim
|       sharedTokens = sharedTokens + 1
|     else
|       unsharedTokens = unsharedTokens + 1
|   end for
|end for
|totalSim = sharedTokensSim / (sharedTokens + unsharedTokens)
|if totalSim < threshold then
|   totalSim = Lin($S_1$, $S_2$)
|return totalSim

4.3 Semantic Similarity (2)

This feature concerned with the case of entities with similar intended meaning, but their descriptions have some “unimportant” or “meaningless” words that affect negatively their similarity. This case mentioned also in [17], but we deal with it differently. This is an example explains this case: an entity in one ontology of the conference domain labeled “conference paper”, on the other hand, ontology of the same domain has this entity labeled “paper”. If the most of other entities in the first ontology have the word “conference” in their descriptions we can say that the word “conference” is unimportant or meaningless word. When we compute the semantic feature mentioned in section 4.2 for these two entities (“conference paper”, “paper”) we find it slightly small, but if we weight the words in terms of their occurrences in the ontology and take these weights into consideration in similarity computation, we get higher and more objective similarity value.

So, to compute this similarity feature, we first need to weight the words and found a function like SoftJaccard one to combine tokens similarities and their weights too.

For weighting the tokens, we consider, as in [17], the information content of a token normalized by the maximum information content as a token weight:

$$weight(t) = \frac{IC(t)}{\max_{i=1,...,T}\{IC(t_i)\}}$$

Where

$$IC(t) = \log\left(\frac{|N|}{|T|}\right)$$

Where $|N|$ : Total number of tokens in the ontology $|T|$ : Occurrences of token $t$ in its ontology

After weighting tokens in each ontology separately, the semantic similarity of two entities $e_1$ and $e_2$ is defined as the max Weighted-SoftJaccard-Lin-Stoilos similarity of their names, labels, and comments:

$$\text{semanticSim2}(e_1, e_2) = \max\{\text{WeightedSoftJaccardLinStoilos}(\text{name}_1, \text{name}_2),\text{WeightedSoftJaccardLinStoilos}(\text{label}_1, \text{label}_2),\text{WeightedSoftJaccardLinStoilos}(\text{comment}_1, \text{comment}_2)\}$$

The detailed algorithm for computing Weighted-SoftJaccard-Lin-Stoilos similarity of two strings $S_1$ and $S_2$ consisted of Tokens$_1$ and Tokens$_2$ with weights vectors $w_1$ and $w_2$ respectively is as follows:

Algorithm 3

|totalSim, sim, weights1, weights2, sharedTokensSim = 0
|for each $t_1$ $\in$ Tokens$_1$, do
|   for each $t_2$ $\in$ Tokens$_2$, do
|     if $t_1$, $t_2$ in WordNet then
|       sim = Lin($t_1$, $t_2$)
|     else
|       sim = Stoilos($t_1$, $t_2$)
|     if Sim > threshold then
|       sharedTokensSim = sharedTokensSim + sim $\ast$ $(w_1 + w_2)$
|       weights2 = weights2 + $w_2$
|     end for
|   end for
|totalSim1 = weights1 + $w_1$
|end for
|totalSim = sharedTokensSim / (weightsSum1 + weightsSum2)
|return totalSim

4.4 Abbreviation Similarity

All the above features fail when the name or label of one entity contains an abbreviation versus multiple words in other entity’s name or label. For example, “PC member” “program committee member” the token “PC” in the former is abbreviation of the tokens “program” and “committee” in the other. We design this feature especially to compute similarity in case of abbreviations.

Abbreviation similarity of two entities $e_1$ and $e_2$ is the max Abbreviation similarity of their names and labels:

$$\text{abbreviationSim}(e_1, e_2) = \max\{\text{Abbreviation}(\text{name}_1, \text{name}_2),\text{Abbreviation}(\text{label}_1, \text{label}_2)\}$$

The detailed algorithm for computing similarity of two strings $S_1$ and $S_2$ is as follows:
4.5 Context similarity

This feature is concerned with measuring similarity in case the entities have non-syntactic and non-sematic similar names or non-expressive ones, such as: “paper” versus “contribution” in the conference domain, or “dc123” versus “document”. In this case we utilize the context similarity idea that had been proposed by [4], but we make some improvement. We define two representations of an entity as a document, one just describes the entity (called entity’s profile) and the other describes the entity with its neighbors (called entity’s context), then compare entities using these two representations, and select the most similar one. The idea behind making those two representations of entity is that sometimes two entities have very similar descriptions but differ in their context which affects negatively their similarity.

We define entity’s profile and entity’s context as follow:

Entity’s profile is a set of words describing the metadata of entity e:

\[ \text{Profile}(e) = \{w_i\} \]
\[ i \in [1, N_m], \text{words occurring in the metadata of } e \]
where \( e \in C \cup P \)

Entity’s context: here we define Concept’s context and Property’s context separately as follow:

Concept’s context of concept c is its profile , its subconcepts’ profiles and profiles of entities used to restrict it:

\[ \text{Context}(c) = \text{Profile}(c) + \text{Profile}(\text{Heir}(c)) \]
\[ + \text{Profile}(\text{Rest}(c)) \]

Property’s context of property is its profile, its subproperties’ profiles and the profiles of concepts in its domain and range:

\[ \text{Context}(p) = \text{Profile}(p) + \text{Profile}(\text{Heir}(p)) \]
\[ + \text{Profile}(\text{Domain}(p)) + \text{Profile}(\text{Range}(p)) \]

After building term-document matrices for each representation (profile and context), we can compare two entities using cosine similarity of their representative vectors:

\[ \text{ContextSim}(e_1, e_2) = \cos(V_c(e_1), V_c(e_2)) \]
\[ \text{ProfileSim}(e_1, e_2) = \cos(V_p(e_1), V_p(e_2)) \]

Where: \( V_c(e) \) and \( V_p(e) \) are the vectors representing e in the term-document matrix of the contexts and profiles respectively.

Now, the IR similarity of two entities \( e_1 \) and \( e_2 \) is the max similarity of their context and profile:

\[ \text{IRSIm}(e_1, e_2) = \max\{\text{ContextSim}(e_1, e_2), \text{ProfileSim}(e_1, e_2)\} \]

5. Training

To train our system, we used samples from Conference and Benchmark tracks of OAEI\(^1\) 2015. These samples are triples (source ontology, target ontology, reference alignment) listed in Table 1. It contains 38318 training example, 444 of them are positive examples (pairs of matched entities i.e. entity pairs that are existed in the reference alignment) and 37874 are negatives (pairs of unmatched entities i.e. entity pairs that are not existed in the reference alignment). Whereas the remaining triples from the conference track was used for testing.

<table>
<thead>
<tr>
<th>Source ontology</th>
<th>Target ontology</th>
<th>Reference alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Conference Track</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cmt.owl</td>
<td>conference.owl</td>
<td>cmt-conference.rdf</td>
</tr>
<tr>
<td>conference.owl</td>
<td>iasted.owl</td>
<td>conference-iasted.rdf</td>
</tr>
<tr>
<td>edas.owl</td>
<td>ekaw.owl</td>
<td>edas-ekaw.rdf</td>
</tr>
<tr>
<td>From Benchmark Track</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101.owl</td>
<td>204.owl</td>
<td>204.rdf</td>
</tr>
<tr>
<td>101.owl</td>
<td>205.owl</td>
<td>205.rdf</td>
</tr>
<tr>
<td>101.owl</td>
<td>238.owl</td>
<td>238.rdf</td>
</tr>
<tr>
<td>101.owl</td>
<td>301.owl</td>
<td>301.rdf</td>
</tr>
<tr>
<td>101.owl</td>
<td>304.owl</td>
<td>304.rdf</td>
</tr>
</tbody>
</table>

Table 1. Training Data

We experimented many Regression techniques: Neural Networks (Multilayer Perceptron), Decision Trees (REPTree), and Rule-based Techniques (M5Rules). We found that the difference between mean squared errors resulted from 10-fold cross validation [18] for these models are not significant as listed in Table 2.

Table 2. MSE of various regression models using 10-fold cross validation

<table>
<thead>
<tr>
<th>Regression model</th>
<th>Mean Squared Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multilayer Perceptron</td>
<td>0.046</td>
</tr>
<tr>
<td>REPTree</td>
<td>0.047</td>
</tr>
<tr>
<td>M5Rules</td>
<td>0.045</td>
</tr>
</tbody>
</table>

Figure 1. F-measure versus threshold for various regression models using separated test set

Table 3. Precision and recall of various models at the best threshold

<table>
<thead>
<tr>
<th>Regression Model</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multilayer Perceptron</td>
<td>0.71</td>
<td>0.63</td>
<td>0.67</td>
<td>160</td>
<td>63</td>
<td>93</td>
</tr>
<tr>
<td>REPTree</td>
<td>0.61</td>
<td>0.69</td>
<td>0.65</td>
<td>175</td>
<td>112</td>
<td>78</td>
</tr>
<tr>
<td>M5Rules</td>
<td>0.68</td>
<td>0.62</td>
<td>0.65</td>
<td>157</td>
<td>71</td>
<td>96</td>
</tr>
</tbody>
</table>

In General we notice that MLP model is the best in terms of F-measure, but by selecting the best threshold for each model and comparing recall values we found that REPTree model has the best recall value as listed in Table 3.

6. Finding Alignment

The trained model (REPTree) is used to predict the similarity value of each entity pair. From these similarities we will decide which pairs are matched, i.e. finding the alignment of the input ontologies. To do so we first applied the naive descending algorithm [19] on the similarity matrix resulted from the regression model and got one-to-one alignment. Then we applied semantic verification technique proposed by ASMOV ontology matching system [6]. That is by applying the patterns showed in Figure 2 to verify that certain axioms inferred from the alignment are actually asserted in ontology and remove correspondences that lead to inferences that cannot be verified.

Figure 2. Semantic verification patterns [6]
7. Evaluation

To evaluate our approach, we designed 4 experiments. Each answers one of the questions we are interested in:

1. Is the REPTree-based combination method better than individual similarity features?
2. Is the REPTree-based combination method better than other combination methods discussed in section 2?
3. Does the semantic verification technique improve REPTree-based ontology matcher? If it does, how much improvement does it make?
4. How does our approach perform compared with others?

7.1 Comparison of REPTree-based combination method with individual similarity features

We proposed five kinds of similarities as features to build our regression model, namely, syntactic similarity, semantic similarity (1), semantic similarity (2), abbreviation similarity, and context similarity. Each feature can be used to find the correspondence between two ontologies from different perspectives. Figure 3 compares the performance (i.e., F-measure at various thresholds) of 5 individual features and the performance of their combination using REPTree-based regression model over OAEI conference test (except the 3 triples used for training, listed in Table 1).

We observe from Figure 3 that our method for combining the five similarity features using the regression model (REPTree) outperforms each individual similarity, i.e. the combination process is effective.

![Figure 3. Performance of individual similarities and the REPTree-based combination method](image)

7.2 Comparison of REPTree-based combination method with other combination methods

Similarity combination has been researched in many ontology matching approaches as discussed in section 2. To evaluate the REPTree-based combination method, we conduct this experiment to compare it with the best two combination methods: Harmonic Adaptive Weighted Sum [7] and local Confidence Weighted Sum [12] as they outperform other combination methods.

The experiment methodology is: For each test, we first calculate five individual similarities (i.e. syntactic similarity, semantic similarity (1), semantic similarity (2), abbreviation similarity, and context similarity) as described in section 4. Then we combine the individual similarities using our method (REPTree-based) and the two other methods: Harmonic Adaptive Weighted Sum (Harmony) and local Confidence Weighted Sum (LC). Precision, recall and F-measure of final results on each test at various thresholds are calculated. Finally the overall precision, recall and F-measure are calculated over all conference tests at various thresholds, which are shown in Figure 4 (a).

We observe from Figure 4 (a) that REPTree-based similarity combination method generally outperforms the two other methods. By viewing precision, recall, and F-measure of the three methods at its best threshold as shown in Figure 4 (b), we notice REPTree-based method holds the highest recall, and F-measure at .69 and .65 respectively.

7.3 The effect of semantic verification

We conduct this experiment to show the improvement of performance gained by applying semantic filtering patterns described in section 6. The experiment methodology is: For each test, we first calculate the five individual similarities as described in section 4. Then we combine
the individual similarities using our method REPTree-based. Precision, recall and f-measure of the resulted alignment on each test at various thresholds are calculated. Also, we calculate Precision, recall and f-measure of resulted alignment after applying naïve descending algorithm and semantic verification patterns on each test at various thresholds. Finally the overall precision, recall and f-measure are calculated over all conference tests at various thresholds for both cases (with and without extraction and semantic verification), which are shown in Figure 5.

We observe from Figure 5 the significant improvement achieved by applying semantic verification patterns for filtering logically wrong correspondences.

7.4 Comparison of REPTree-based Matching system with other matching systems
We called our matching system ML-Matcher. It uses regression learning model (REPTree), described in section 5, to predict pairwise similarities from the five similarity features described in section 4, then extracts alignment using naïve descending algorithm and apply semantic verification patterns showed in Figure 2.

Figure 6 compares the performance of ML-Matcher and 10 top-ranked ontology matching systems on the conference tests (except the 3 triples used for training, listed in Table 1) at OAEI campaign 2015 [20]. The evaluation data of these 10 systems can be downloaded here\(^2\). The data of ML-Matcher can be downloaded here\(^3\).

The results in Figure 6 show that ML-Matcher holds the 3\(^{rd}\) top position in terms of F-measure and the 2nd position in terms of recall. The recall of ML-Matcher (0.69565) is competitive to AML (0.6996) and outperforms others.

---

\(^2\) [Link](http://oaei.ontologymatching.org/2015/conference/data/conference2015-results.zip)

\(^3\) [Link](https://www.dropbox.com/s/dmax7exrq2aldao/ML-Matcher%20results.rar?dl=0)
8. Conclusion

In this paper we proposed a new ontology matching approach, the ML-Matcher. First the ML-Matcher measures five different similarities for each entity pair in the input ontologies. Then it predicts the overall similarity by using the learned regression model (REPTree). Later the naïve descending algorithm is used to extract the alignment. Finally semantic verification patterns are used to remove semantically unverified correspondences.

The proposed method for estimating the overall similarity value of two entity pair is efficient because it adopts different similarity measures, which are well designed to capture different kinds of terminological heterogeneity, and combines their results using a regression model which makes it possible to discover more correspondences by producing higher and more objective similarity values.

Using the semantic verification proposed by ASMOV to remove incorrect correspondences significantly improves the performance of the matching results produced by the proposed regression model.

ML-Matcher is competitive with the 3 top-ranked systems on the conference tests at OAEI campaign 2015.

Future work may include: Explore and implement more similarity features that depend on ontology structure. Investigate more semantic patterns to add correspondences through exploiting ontology axioms.

References


