DOME Results for OAEI 2019

Sven Hertling[0000−0003−0333−5888] and Heiko Paulheim[0000−0003−4386−8195]

Data and Web Science Group, University of Mannheim, Germany
{sven,heiko}@informatik.uni-mannheim.de

Abstract. DOME (Deep Ontology MatchEr) is a scalable matcher for instance and schema matching which relies on large texts describing the ontological concepts. The doc2vec approach is used to generate a vector representation of the concepts based on the textual information contained in literals. The cosine distance between two concepts in the embedding space is used as a confidence value. In comparison to the previous version of DOME it uses an instance based class matching approach. Due to its high scalability, it can also produce results in the largebio track of OAEI and can be applied to very large knowledge graphs. The results look promising if huge texts are available, but there is still a lot of room for improvement.

Keywords: Ontology Matching · Knowledge Graph · Doc2Vec

1 Presentation of the system

Ontology matching is a key feature for the semantic web vision because it allows to use and interpret datasets which are unknown at the time of writing knowledge accessing software. [11] shows that there are many different elementary matching approaches on element, structure and instance levels. The `Deep Ontology MatchEr` (DOME) focuses at element and instance level matching. One of the reasons is that there are more and more instance matching tracks at the OAEI (Ontology Alignment Evaluation Initiative) like `SPIMBENCH`, `Link Discovery`, and `Knowledge graph`. These tracks need a scalable matching system. Thus, the main signal for finding correspondences is string based. Many other knowledge graphs in the Linked Open Data Cloud [2] also have a lot of literals with long texts which can be optimally used by the matching framework presented in this paper. Especially knowledge graphs extracted from Wikipedia such as DBpedia [1] or YAGO [5] contains descriptions of resources (abstracts of wiki pages).

1.1 State, purpose, general statement

The overall matching strategy of DOME is shown in figure 1. It starts with a simple string matching followed by a confidence adjustment. This is applied for

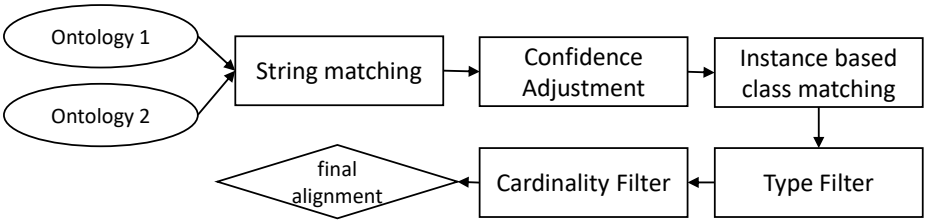

Fig. 1. Overview of the DOME matching strategy.

all classes, instances, and properties. The latter one includes *owl:ObjectProperty*, *owl:DatatypeProperty*, and *rdf:Property* (as retrived by the jena[1] method *OntModel.listAllOntProperties()*). As a next step in the pipeline, an instance based class matching is applied. It uses all matched individuals and based on those types, tries to find meaningful class mappings.

The following type filter deletes all correspondences where the type of source and target concept is different (like *owl:DatatypeProperty - owl:ObjectProperty*). This might happen because all properties (also *rdf:Property*) can be matched with each other. The final cardinality filter ensures a one to one mapping by sorting the correspondences by confidence and iterates over them in descending order. If the source or target entity is not already matched, it counts a valid correspondence - otherwise it will be dropped and will not appear in the final alignment.

In the following, the first three matching stages of DOME are discussed in more detail.

String matching As shown in figure 2, DOME uses multiple properties for matching all types of resources. If a *rdfs:label* from ontology A matches the *rdfs:label* from a resources in ontology B after the preprocessing, DOME creates a mapping with a static confidence of 1.0. The same confidence is applied when a *skos:prefLabel* matches. In case a URI fragment or *skos:altLabel* fits, a lower confidence of 0.9 is used.

The string preprocessing consists of tokenizing the text (also takes care of CamelCase[2] formatting), stopword removal and lowercasing. Afterwards the text is concatenated together to form a new textual representation. In case the initial text contains mostly numbers, the whole text is discarded.

Confidence Adjustment The confidence adjustment stage of DOME iterates over all correspondences and reassign a new confidence in case it is possible. The main approach used here is doc2vec [7] which is based on word2vec [8]. It allows to compare texts of different lengths and represent them as a fixed length vector. A comparison of these vectors can be achieved with a cosine similarity.

[1] `https://jena.apache.org`
[2] `https://en.wikipedia.org/wiki/Camel_case`

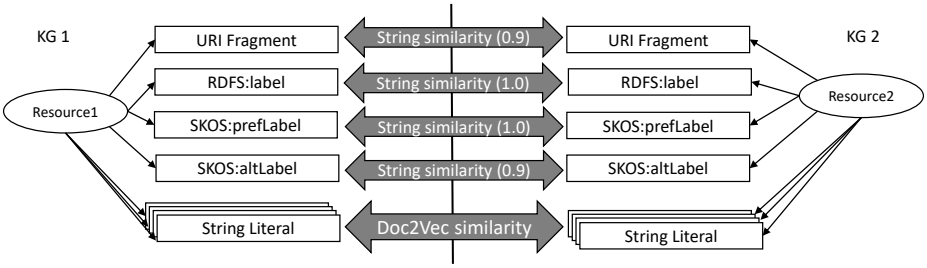

Fig. 2. DOME literal comparisons.

In comparison to DOME submitted to OAEI 2018, the generation of the text for a given resource has changed. In the current version, all statements in the ontology are examined where a given resource has the subject position. If the object is a literal and the datatype of it corresponds to *xsd:string* or *rdf:langString* or contains a language tag, it will be selected. All those literals are preprocessed in the same way as described in paragraph *string matching* and concatenated together. This text forms a document which is used for training a doc2vec model. DOME uses the DM sequence learning algorithm with a vector size of 300 and window size of 5 as in the previous version of this matcher dla[3]. The minimal word frequency is set to one to allow all words contribute to the concept vector. The adjusted confidence is later used in the cardinality filter to create a 1:1 mapping.

Instance based class matching After the class, instance, and property matching an additional class alignment step is performed. The basic idea is to inspect the types (classes) of already matched instances. If two individuals are the same, there is a high probability that some of the corresponding types should be also matched.

We experimented with three different similarity metrics for two given classes c_1 and c_2. The dice similarity metric [9] is defined as follows:

$$Sim_{DICE}(c_1, c_2) = \frac{2 * |I_{c_1} \cap I_{c_2}|}{|I_{c_1}| + |I_{c_2}|} \in [0...1]$$

I_{c_1} and I_{c_2} denotes the set of instances which have c_1 (c_2) as one of its type. $I_{c_1} \cap I_{c_2}$ corresponds to the matched instances which are typed with both c_1 and c_2. Sim_{DICE} corresponds to the overlap of matched instances with both classes and all instances of the two classes separately.

[6] also includes a more relaxed version of the previous similarity called Sim_{MIN} which is defined as

$$Sim_{MIN}(c_1, c_2) = \frac{|I_{c_1} \cap I_{c_2}|}{min(|I_{c_1}|, |I_{c_2}|)} \in [0...1]$$

It interrelates the matched instances with both classes and the instances of the smaller-sized class. As stated in [6] Sim_{DICE} is always smaller or equal to Sim_{MIN}.

A third possibility is Sim_{BASE} [6] which matches the classes c_1 and c_2 in case at least one instance with those classes is matched:

$$Sim_{BASE}(c_1, c_2) = \begin{cases} 1 & \text{if } |I_{c_1} \cap I_{c_2}| > 0 \\ 0 & \text{if } |I_{c_1} \cap I_{c_2}| = 0 \end{cases} \in [0...1]$$

After experimenting with those measures, it turned out that Sim_{BASE} introduces a lot of wrong correspondences because each error in the instance matching is directly forwarded to the class matches. Sim_{MIN} needed a very low threshold and ranks the classes suboptimal. Thus some similarity between Sim_{BASE} and Sim_{MIN} is needed. One possible way is to incorporate the quality of the matcher at hand - especially how many instance correspondences it finds. Thus another similarity called Sim_{MATCH} is used in DOME and defined as follows:

$$Sim_{MATCH}(c_1, c_2) = \frac{|I_{c_1} \cap I_{c_2}|}{|C_I|} \in [0...1]$$

where C_I represents all instance correspondences created by the matcher. The threshold is set to 0.01 meaning that 1 % of the matches should have the same class. If this is the case, the classes will be matched with a confidence of $Sim_{MATCH}(c_1, c_2)$. This value is rather low. All correspondences generated by this step are therefore scaled to minimum of 0.1 and maximum of 1.0.

1.2 Specific techniques used

The two main techniques used in DOME are the doc2vec approach [7] for comparing the textual representation of the resources and the instance based class matching component.

1.3 Adaptations made for the evaluation

As in the previous version of DOME for OAEI 2018 the DL4J[3] (Deep Learning for Java) library is used as an implementation of the doc2vec approach. Running DOME with this dependency is not easy in SEALS. Therefore we use MELT[4] to package our matcher. The framework generates an intermediate matcher which executes an external process (which is again in Java). This process runs now in its own Java virtual machine (JVM) and allows to load system dependent library files (files with *dll* or *so* extension). [3] explains in more detail why this is necessary.

1.4 Link to the system and parameters file

DOME can be downloaded from
`https://www.dropbox.com/s/1bpektuvcsbk5ph/DOME.zip?dl=0`.

[3] `https://deeplearning4j.org`

2 Results

This section discusses the results of DOME for each track of OAEI 2019 where the matcher is able to produce results. The following tracks are included: anatomy, conference, largebio, phenotype, and knowledge graph track.

Similar to the previous version of DOME, the current matcher is not able to match multiple languages and thus fail on multifarm track. Specific interfaces and matching strategies for the complex and interactive track are currently not implemented.

2.1 Anatomy

For the anatomy track, DOME uses the string comparison method which results in similar precision and recall as the baseline. Properties like *oboInOwl:hasRelatedSynonym* or *oboInOwl:hasDefinition* are used to generate a textual representation of the concepts but this does not introduce better confidence values.

DOME returns 948 correspondences. 932 matches with a confidence of 1.0 which are all correct. 12 correspondences scored with 0.9 are all false positives. Therefore a confidence filter would make sense for this specific track.

The presented matcher has a very low runtime and scales to very huge ontologies. The runtime of 23 seconds is the second best value in this track.

Due to a slightly lower recall (0.007) and precision (0.001) DOME has a lower F-Measure (0.006) than the baseline. The reason could be the different string preprocessing techniques.

2.2 Conference

In the following analysis we refer to the `rar2`[4] reference alignment because it contains more correspondences which are carefully resolved by an evaluator.

When matching classes DOME is same as the edna baseline. Most correspondences have a confidence of 0.9 because the conference track has mostly all textual information in URL fragments. Only one mapping is scored with 1.0 which is <*edas:Country*, *iasted:Conference_state*, =, 1.0>. It is generated by the instance based class matching because both contain *Mexico* as an individual. This mapping is a false positive. The instance based class matching could not help here, because in most of the test cases no instances are available. Properties are matched with an F1-measure of 0.22 which is better than the edna baseline but lower than 5 other matchers. In comparison to the old version of this matcher, the F1-measure is increased by 0.01. Figure 3 shows the result of DOME divided into test cases. It shows that in four test cases (where the source ontology is *confOf*) the matcher is not able to return true positive correspondences.

[4] `http://oaei.ontologymatching.org/2019/results/conference/index.html`

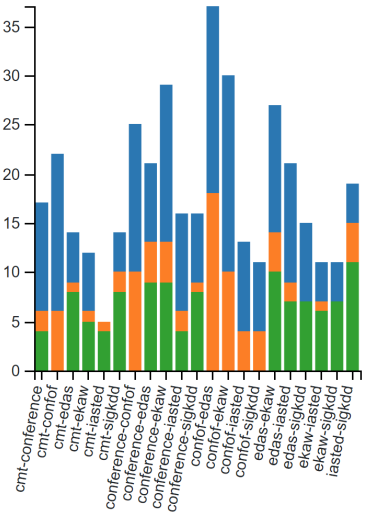

Fig. 3. Analysis of results for conference track. The x axis represents the test cases and y axis the amount of correspondences. Green bars indicates true positives, orange bars false positives, and blue bars false negatives. The plot is generated by MELT framework [4].

2.3 Largebio

As the name already suggests, the largebio track needs matchers which scale well. Test case four is a large test case which matches the whole FMA ontology with a large fragment of SNOMED. The source ontology has 78,989 classes and the target ontology 122,464 classes. This would result in more than 9 billion comparisons when doing it naively. The runtime of DOME for this test case is 38 seconds which is the second best runtime. Moreover DOME is able to complete all tasks within the given timeout.

In task 3, 4, 5, and 6 DOME has the highest precision of all matchers but misses a lot of correspondences in the gold standard and has therefore a lower recall. In task one and two matcher Wiktionary have a higher precision. F-measure wise DOME usually beats Wiktionary and AGM but AML and LogMap variants are better.

2.4 Phenotype

In phenotype track, the matcher should find alignments between disease and phenotype ontologies. The matcher has the highest precision of 0.997 together with FCAMapKG for test case HP-MP and second best for task DOID-ORDO. With the low recall of 0.303 and 0.426 the F-measure is around 0.465 and 0.596.

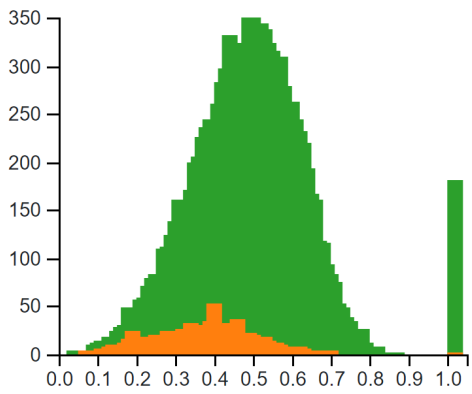

Fig. 4. Analysis of confidences in knowledge graph track. The x axis represents the confidence value and the y axis shows the amount of correspondences. Green bars indicates true positives and orange bars indicates false positives. False negatives are left out because they don't have any confidence assigned by the matching system. The plot is generated by MELT framework [4].

2.5 Knowledge Graph

In the second version of the knowledge graph track, the systems should be able to match classes, properties and instances. DOME was able to run 4 out of 5 test cases. The remaining test case could not be finished because of memory issues.

In comparison to the previous version of the track, classes are more difficult to match. DOME could achive an F-measure of 0.77 for classes (not counting the unfinished test case) and 0.96 for properties. Only FCAMap-KG and Wiktionary are better in matching the latter one. Instances are matched with a F-measure of 0.88 (again not counting the unfinished test case). In average DOME returns 22 class, 75 property, and 4,895 instance mappings.

3 General comments

3.1 Comments on the results

The discussion of the results shows that DOME is in a development phase. Some improvements are already incorporated and some further ideas are discussed in the next section.

3.2 Discussions on the way to improve the proposed system

One further improvment is still the ability to match different languages. As stated in [3] we could use cross lingual embeddings as shown in [10]. Another possibility would be to use a translation step in between.

The confidence adjustment step can not only be done with doc2vec based models but also with tf-idf or other document comparison methods. This should be tried out in future version of this matcher.

The memory issue in the knowledge graph track can be solved by writing all text representations of all resources on disk and train the doc2vec model on this file.

4 Conclusions

In this paper, we have analyzed the results of DOME in OAEI 2019. It shows that DOME is a highly scalable matcher which generates class, property and instance alignments. With the new component DOME is able to match classes based on instances and thus increase the recall of class alignments.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735. Springer (2007)
2. Bizer, C., Heath, T., Idehen, K., Berners-Lee, T.: Linked data on the web (ldow2008). In: Proceedings of the 17th international conference on World Wide Web. pp. 1265–1266. ACM (2008)
3. Hertling, S., Paulheim, H.: Dome results for oaei 2018. In: OM@ ISWC. pp. 144–151 (2018)
4. Hertling, S., Portisch, J., Paulheim, H.: Melt - matching evaluation toolkit. In: SEMANTICS. Karlsruhe. (2019)
5. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: Yago2: A spatially and temporally enhanced knowledge base from wikipedia. Artificial Intelligence **194**, 28–61 (2013)
6. Kirsten, T., Thor, A., Rahm, E.: Instance-based matching of large life science ontologies. In: International Conference on Data Integration in the Life Sciences. pp. 172–187. Springer (2007)
7. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning. pp. 1188–1196 (2014)
8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
9. van Rijsbergen, C.: Information retrieval (1979)
10. Ruder, S., Vulić, I., Søgaard, A.: A survey of cross-lingual word embedding models. arXiv preprint arXiv:1706.04902 (2017)
11. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV, Lecture Notes in Computer Science, vol. 3730, pp. 146–171. Springer Berlin Heidelberg (2005)