DOME results for OAEI 2018

Sven Hertling and Heiko Paulheim

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

Abstract. DOME (Deep Ontology MatchEr) is a scalable matcher which relies on large texts describing the ontological concepts. Using the doc2vec approach, these texts are used to train a fixed-length vector representation of the concepts. Mappings are generated if two concepts are close to each other in the resulting vector space. If no large texts are available, DOME falls back to a string based matching technique. Due to its high scalability, it can also produce results in the largebio track of OAEI and can be applied to very large ontologies. The results look promising if huge texts are available, but there is still a lot of room for improvement.

1 Presentation of the system

1.1 State, purpose, general statement

Ontology matching is often based on string comparisons because each resource is described by URI fragments (last part of an URI after the # sign), `rdfs:label`s, and `rdfs:comment`s. The DOME matcher specifically relies on large texts which describes the resources, and thereby allows to make a better distinction in case of a similar labels. Especially in knowledge graphs like DBpedia or YAGO, such texts are easily extracted from the corresponding Wikipedia abstract.

The usual problem with such large texts is the matching with other similar and long texts. One possible way is to use topic modeling like latent semantic analysis(LSA [2]) or latent dirichlet allocation (LDA [1]). The extracted topics can then be used to find overlaps and in the end similar concepts.

DOME uses another approach called doc2vec (also paragraph vector [5]) which is based on word2vec [6]. The idea is to represent a variable-length texts, like sentences, paragraphs, and documents, as a fixed-length feature vector. This vector is trained to predict the words appearing in the document. Thus this vector represents the semantics of the concept when training on texts which defines the meaning of the concept.

Two approaches for training this vector are established: Distributed Memory (DM) and Distributed Bag of Words (DBOW). Applied to an example concept like Harry Potter[1] the framework of DM is shown in figure 1. During training, the algorithm iterates over the given text in a sliding window of a specified and fixed length. The goal is to predict the last word given the first n words. One

[1] `http://harrypotter.wikia.com/wiki/Harry_Potter`

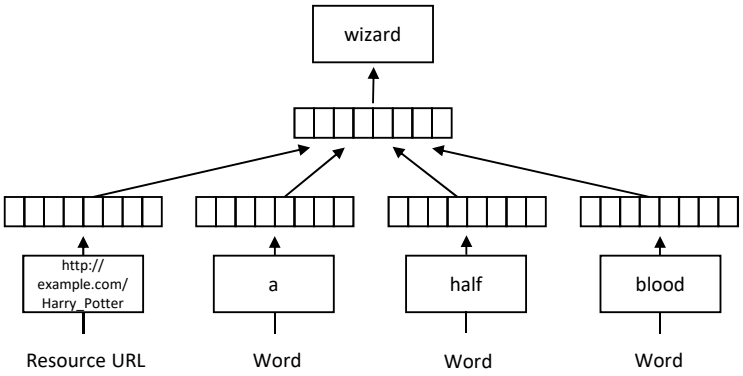

Fig. 1. Training of Distributed Memory given the concept Harry Potter and a small excerpt of the corresponding wiki abstract.

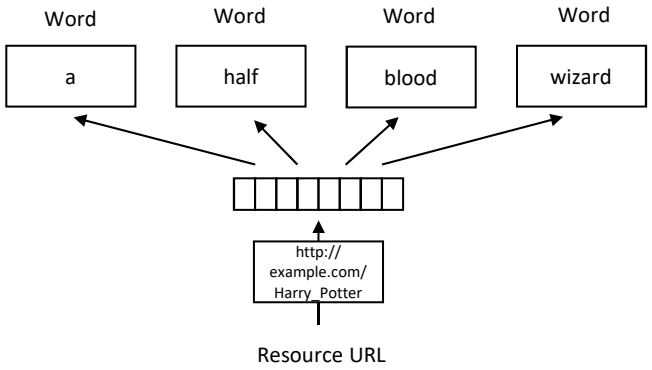

Fig. 2. Training of Distributed Bag of Words. The example is the same as in figure 1 but now the concept URI together with a small subset of text is usbed to predict the following word.

special vector is the first one which represents the paragraph vector. In our case this is the URI of the concept. All large texts which define this resource can be used to train this vector.

Another approach for generating the concept vector is Distributed Bag of Words (DBOW), shown in figure 2. Instead of using concept vectors for each word, it tries to predict words from the text as an output.

DOME uses the DM sequence learning algorithm with a vector size of 300 and window size of 5. The training is repeated in 10 epochs. The minimal word frequency is set to the minimum to allow all words contribute to the concept vector. We compute a predefined set of properties which contains definitional texts by two simple rules: 1) directly choose `rdfs:comment` 2) use every property where the URI ends in "abstract". This can be further improved in the next version of DOME.

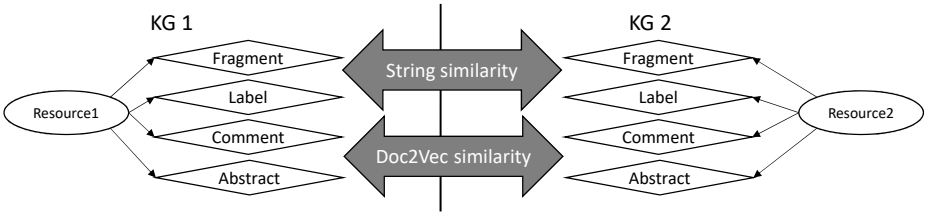

Fig. 3. Matching strategy of DOME

The doc2vec model is trained on all texts available in both ontologies. For each concept in the second ontology, the corresponding concept vector is computed, and the concepts which have the most similar vectors to those from the first ontology are retrieved. A mapping between two resource is established when the cosine similarity is above 0.85 (the threshold has been chosen based on a manual inspection of the results).

The whole matching approach is shown in figure 3. The labels and fragments of each resource are compared using string based similarity. Specifically, the texts are tokenized and all punctuation (especially underscores and the like) are removed. After lowercasing, these values are stored in a hash structure. A mapping is created when each fragment or label have an exact match. The confidence value of these alignments are set to 1.0. After this step, the doc2vec approach is applied to find further matching concepts. We ensured that the mapping is OWL compliant because we only match instances to instances, classes to classes, and properties to properties. In the latter case we further distinguish datatype properties and object properties but also match properties declared as rdf:properties. With such a setup, the matcher is very scalable and can match all types of resources.

1.2 Specific techniques used

The main technique used in DOME is the doc2vec approach [5] for comments and abstracts of concepts. It is only activated when there is enough text to process. All other matching techniques rely on fast string similarity. Further filtering of the alignment is not executed but during the matching only one to one mappings are allowed.

1.3 Adaptations made for the evaluation

DOME is implemented in java and uses the DL4J[2] (Deep Learning for Java) as an implementation of the doc2vec approach. DL4J heavily relies on platform specific implementations which are stored in multiple JAR files. This allows it to make use of GPUs to further speed up the computation. DOME relies on the

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

CPU implementation of DL4J because upfront it is not clear if all evaluation machines used for OAEI contain a DL4J compatible GPU.

Although the DL4J framework allows for searching for related concepts, it does not provide the similarity values out of the box. Thus the framework is modified to also retrieve these value which can be used in the alignment file to represent the confidence of a mapping. Since the values are already normalized no further post processing step of the similarity values is needed.

Unfortunately, the packaged SEALS matcher was not able to run under the SEALS evaluation routine. The SEALS client loads all JAR files in its own classpath. This is a very secure way of running third-party code, but at the same time one of the most frequent cauess of matchers not working at OAEI, as in the case of DOME. The root cause is the custom classloader of SEALS which uses the JCL library[3]. The SEALS classloader is a subclass of the AbstractClassLoader in the JCL library. Both classloaders do not implement all methods (especially the getPackage method) of the standard classloader. Many other libraries use such functions to further load operating specific code. This applies to the DL4J library as well as the sqlite-jdbc library.

We fixed the error by creating an intermediate matcher which calls another java process. Within that process the classloader is the standard one and the DL4J library could be loaded without any errors. We released a matching framework which does the SEALS and Hobbit packaging, uploading and creating the intermediate matcher.[4]

1.4 Link to the system and parameters file

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

2 Results

The following section discusses the results for each track of the OAEI 2018 where DOME is able to produce meaningful results. This includes the anatomy, conference, largebio, phenotype, and knowledge graph track.

DOME was not able to complete the multifarm track because currently no translation component is included. This would be possible with cross lingual embedding approaches shown in [8]. For complex and interactive track the matching system has to produce different type of output mapping or matching strategy which is not implemented. The tracks biodiv and iimb don't contain enough free texts in the selected properties.

[3] `https://github.com/kamranzafar/JCL`
[4] `https://github.com/sven-h/ontMatchingHobbit`

2.1 Anatomy

In the anatomy track, there are only labels given, thus the doc2vec approach is not used here. There are some properties like `oboInOwl:hasRelatedSynonym` or `oboInOwl:hasDefinition` which point to resources with more describing text, but these resources is not recognized by DOME, since we do not implement a larger list of properties used to point to texts.

Therefore, DOME only utilizes string based matching for this track. The text is lowercased, tokenized and then matched based on a hashing algorithm. This results in a high precision of 0.997 (similar to the string equivalence baseline) and a very low runtime of 22 seconds. Only LogMapLt was 4 seconds faster.

Due to a slightly lower recall of 0.615 (0.07 lower than the baseline) DOME has a lower F-Measure than the baseline.

In improvement in this track would be to use the additional texts from `oboInOwl:hasRelatedSynonym` and `oboInOwl:hasDefinition` to further increase the recall. In order not to have to manually maintain such a list, it would also be possible to incorporate all literals that consist of text of at least a certain number of words.

2.2 Conference

Within the conference track, DOME is as bit better than the baseline and often similar to edna (which is a string editing distance matcher adopted from the benchmark track). Evaluating DOME against the original reference alignment it performs exactly like edna in the class mappings and a bit better in the property mappings - both in terms of recall and precision. This results in 0.07 better F-Measure. But there is room for a lot improvement, because in this year, the best matcher reached 0.58 F-Measure in this track.

When comparing to the entailed reference alignment DOME has same evaluation measures like edna and a bit better when comparing properties. If both classes and properties are taken into account DOME is only 0.01 better than edna and 0.15 behind the current best matcher.

In most of the conference ontologies, there are no long natural language texts. Only in rare cases, some classes are described by a comment. Those were processed by the doc2vec model but does not yield any new mappings.

2.3 Largebio

In the largebio track, the number of classes is very high. In the case of FMA-SNOMED this results in matching 78,989 classes to 122,464 classes. Matchers which compare a string from one ontology to all concepts of the other ontology have a quadratic runtime and usually can not finish in time. DOME is one of five matchers (DOME, FCAMapX, LogMap, LogMapBio, XMap) which were able to return results within the given time limit. It is the fastest one and terminates within 30 seconds on the largest track. The second fastest is XMap with 7 minutes and the slowest one is LogMapBio with 49 minutes. The reason here is the

same as in the anatomy track. Most resources are only described by a label and fragment without further textual content. Thus, DOME relies on string comparison with a high precision but low recall. In case of "SNOMED-NCI whole", this results in a precision of 0.907 and a recall of 0.485 (F-Measure of 0.632). The best matcher on this subtrack in terms of F-Measure is FCAMapX with a value of 0.733.

2.4 Phenotype

The phenotype track is based on a real use case, and the matcher should find alignments between disease and phenotype ontologies. DOME is also able to complete this track but with a low F-Measure of 0.483 (HP-MP) and 0.633 (DOID-ORDO). The precision is again the highest among all matchers, but the recall is below 0.5.

However, some ontologies in this track, like the DOID ontology, have further properties containing describing texts like `obo:IAO_0000115` (label of the property is definition). DOME in its current version does not make use of this property, but, as discussed for the anatomy track above, those could be utilized by extending the system.

2.5 Knowledge Graph

The knowledge graph track is a new track where classes, properties and instances should be matched. As already pointed out in [3,4], matching the classes and properties is easier than the instances. This is also the case for the DOME matcher.

It returns all three types of mappings and complete on all nine sub tasks. In average it returns 16 class, 207 property, and 15,912 instance mappings.

DOME achieved an F-Measure of 0.73 in the class correspondences. It is balanced between recall and precision, but even the baseline has a higher recall. So there should some room for improvement.

When analyzing the property alignments, only DOME and the baseline can produce any results. Most likely, the reason is that all properties are typed as `rdf:Property` and not subdivided into `owl:DatatypeProperty` and `owl:ObjectProperty`. As discussed above, DOME is configured to match also `rdf:Property`. This results in a F-Measure of 0.84.

Instance matches are generated by AML, DOME, LogMap, LogMapLt and the baseline. Especially in the instance mapping the doc2vec approach can help because long comments and abstracts of the resources are available. DOME was the second best matcher with an F-Measure of 0.61 (the baseline is the best "matcher" with an F-Measure of 0.69).

Overall, looking at the results for classes, properties, and instances together, DOME has an F-Measure of 0.68, which is better than all matchers except the baseline.

3 General comments

3.1 Comments on the results

The overall results shows that DOME is in a development phase. Sometimes it can beat at least the baselines in terms of F-measure and sometimes not. Currently there are not many tracks which provide a large amount of describing text for each resource, but many ontologies and knowledges graphs exists out there where this is the case.

3.2 Discussions on the way to improve the proposed system

Based on the evaluation on all kinds of different tracks, we noticed a lot of further improvements. First of all, some ontologies use properties which connect a resource to its describing text which are not recognized by DOME. One possible approach to fix this would be the use all properties which contain long texts by some heuristic, e.g., strings exceeding a certain number of characters on average. This would include more text to help the doc2vec model to better differentiate the concepts.

Another possible improvement is to use pretrained word vectors. Those might contain more semantics for each word than training it directly on describing texts for the two ontologies. However, for some very domain-specific ontologies with large amounts of texts, the generic pre-trained embeddings might even perform worse, thus, it is an open research question which of the two yields better results.

A third possible approach is to combine the approach of RDF2Vec [7] (i.e., computing the word2vec embedding of random walks within knowledge graphs) and various cross lingual embeddings shown in [8]. One simple approach would be to learn a linear transformation between the two generated embeddings of the ontologies.

4 Conclusions

In this paper, we have introduced the DOME matcher, which relies on document embeddings for texts describing the concepts defined in an ontology. The results for DOME are analyzed on the different tracks of OAEI. DOME is a highly scale matching system capable of generating class, property and instance alignments. On some tracks where a lot of text describing each resource exists, it shows promising results. However, the matcher is currently in an early state and offers a lot of room for improvement.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research 3(Jan), 993–1022 (2003)
2. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American society for information science 41(6), 391–407 (1990)
3. Hertling, S., Paulheim, H.: Dbkwik: A consolidated knowledge graph from thousands of wikis. In: IEEE International Conference on Big Knowledge, ICBK 2018, Singapore (2018)
4. Hofmann, A., Perchani, S., Portisch, J., Hertling, S., Paulheim, H.: Dbkwik: Towards knowledge graph creation from thousands of wikis. In: Proceedings of the International Semantic Web Conference (Posters and Demos), Vienna, Austria. pp. 21–25 (2017)
5. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning. pp. 1188–1196 (2014)
6. 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)
7. Ristoski, P., Paulheim, H.: Rdf2vec: Rdf graph embeddings for data mining. In: International Semantic Web Conference. pp. 498–514. Springer (2016)
8. Ruder, S., Vulić, I., Søgaard, A.: A survey of cross-lingual word embedding models. arXiv preprint arXiv:1706.04902 (2017)