DOME results for OAEI 2018

Sven Hertling

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

Abstract. DOME (Deep Ontology MatchEr) is a scalable matcher which relies on large texts describing the ontological concepts. With the help of 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 two each other in the resulting high dimensional vector space. If no large texts are available DOME falls back to a string based matching technique. Due to the high scalability it can also return results in the largebio track of OAEI and can be applied to very large ontologies. The results look promising when 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:labels and rdfs:comments. The DOME matcher specifically relies on large texts which describes the resource. It allows to better make a distinction in case of a similar label. Especially in knowledge graphs like DBpedia or YAGO such texts are easily extracted from the correspondent 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 text like sentences, paragraphs and documents with 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

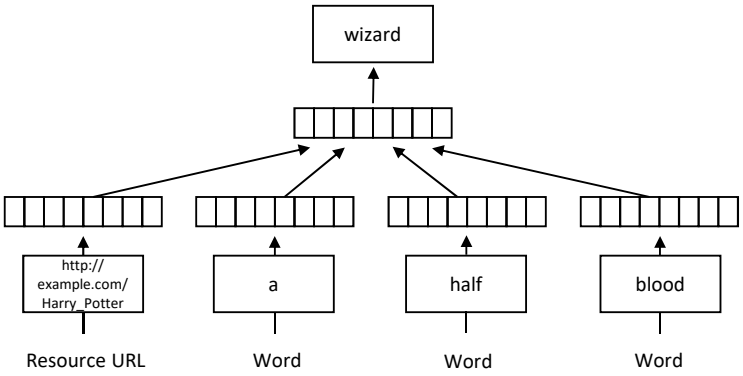

Fig. 1. Training of Distributed Bag of Words given the concept Harry Potter and a small excerpt of the corresponding wiki abstract.

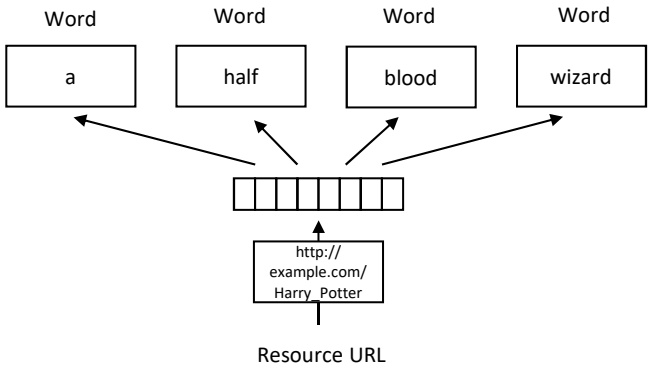

Fig. 2. Training of Distributed Bag of Words. The example is the same as in figure 1 but now the concept URI together with a small subset of text is usbed to predict the following word.

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 defines this resource can be used to train this vector.

Another framework for retrieving the concept vector is Distributed Bag of Words (DBOW) shown in figure 2. It ignores the concept vector in the input. Instead 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 äbstract:̈ This can be further improved in the next version of DOME.

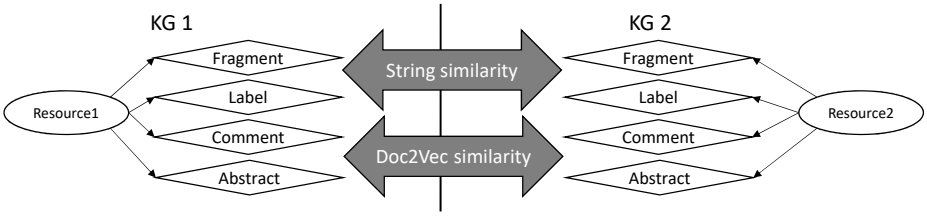

Fig. 3. Matching strategy of DOME

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 most similar vectors from the first ontology are retrieved. A mapping between two resource is made when the cosine similarity is above 0.85 (the value is chosen based on a manual inspection of the results). The DL4J framework do not provide these similarities 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.

The whole matching approach is shown in figure 3. The labels and fragments of each resource are compared using string based similarity. In more detail 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 this alignment is set to 1.0. After this step the doc2vec approach is applied to further find similar concepts. We ensured that the mappings 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 an 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) framework to have a stable implementation of the doc2vec approach. DL4J heavily relies on platform specific implementations which are stored in multiple jars.

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

This allows it to make use of the graphic card to further speed up the computation. DOME relies on the CPU implementation of DL4J because upfront it is not clear if all evaluation machines contains a DL4J compatible graphic card.

Unfortunately, the packaged SEALS matcher was not able to run under the SEALS evaluation routine. The SEALS client loads all jars in its own classpath. This might be also an security issue but is most often the cause of not working matchers 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.

2.1 Anatomy

In the anatomy track there are only labels given and thus the doc2vec approach is not used here. There are some properties like oboInOwl:hasRelatedSynonym or oboInOwl:hasDefinition which points to resources with more describing text but these resources is not recognized by DOME.

Thus only string based matching on the label is performed. 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 little bit 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 hasRelatedSynonym and hasDefinition to further increase the recall.

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

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 room for a lot improvement because in this year matcher reach 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. In rare cases some classes are described by a comment. This is processed by the doc2vec model but does not yield any new mappings.

2.3 Largebio

In largebio track the number of classes are 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 S̈NOMED-NCI wholeẗhis 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 blow 0.5.

For example the DOID ontology have further properties with describing texts like obo:IAO_0000115 (label of the property is definition). DOME in its current version does not make use of this property.

2.5 Knowledge Graph

The knowledge graph track is a new track where classes, properties and instances should be matched. As pointed out in [4] and [3] 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.

The presented matcher achieved a 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 results. Most probably the reason is that all properties are typed as rdf:Property and not subdivided into owl:DatatypeProperty and owl:ObjectProperty. Thus DOME matches datatypeProperty to datatypeProperty and objectProperty to objectProperty but in the end also matches any rdf:Property. This resulted in a F-Measure of 0.84.

The instance matching is done 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 have an F-Measure of 0.69).

If no differentiation between classes, properties and instances are made then 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 provides huge 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 e.g. ontologies we noticed a lot of further improvements. First of all there are sometimes properties which connects the resource to a describing text which are not recognized by DOME. One way out would be use all properties which contains long texts e.g. more than a fixed number of characters. This would include more text to help the doc2vec model to better differentiate the concepts.

Another possibly for improvement is to use pretrained word vectors. They might contain more semantics for each word than training it directly on the two ontologies.

A third way is to combine the approach of RDF2Vec [7] (comuting 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 the results for DOME are analyzed. It is a highly scale matching system capable of generating class, property and instance alignments. The matcher is currently in an early state and multiple improvements are already identified. Nevertheless on some tracks with a lot of text describing each resource it shows better results.

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)