OntoConnect: Results for OAEI 2020

Jaydeep Chakraborty[1], Beyza Yaman[2], Luca Virgili[3], Krishanu Konar[4], and Srividya K. Bansal[1]

[1] CIDSE, Arizona State University, Tempe, Arizona
[2] ADAPT Centre, Dublin City University, Dublin, Ireland
[3] Polytechnic University of Marche, Ancona, Italy
[4] Media.net, Mumbai, India

Abstract. The results of OntoConnect, an Ontology alignment system, in the Ontology Alignment Evaluation Initiative (OAEI) 2020 campaign is reported in this paper. OntoConnect is a domain-independent schema alignment system that combines syntactic similarity and structural similarity between classes/concepts to align the classes/concepts from the source and target ontologies. This paper describes the participation of OntoConnect at OAEI 2020 and discusses its methodology and results on the Anatomy dataset.

Keywords: Ontology alignment · Ontology Matching · Unsupervised Learning · Recursive Neural Network.

1 Presentation of the system

OntoConnect [3] is an ontology alignment system that uses an unsupervised machine learning technique that can predict similar source and target ontology classes based on their ontological structure (hierarchy, meta-information, etc.) and syntactic structure without any background domain knowledge or domain expert intervention in contrast to existing learning-based approaches. In the following sections, we present the methodology behind the system and the results of the system participation in the OAEI initiative.

1.1 State, purpose, general statement

Ontology alignment is a process to integrate multiple knowledge bases to eliminate data heterogeneity. There are many ways to address the ontology alignment problem such as string-based approach, language-based approach, semantic approach, extensional approach, etc. Most of the current state-of-the-art ontology alignment systems depend on domain knowledge that makes the alignment process domain-specific, time-consuming, and error-prone to human error. To overcome this challenge, we developed an ontology alignment approach that is independent of domain knowledge and does not need the domain expert intervention. In this paper, the OntoConnect ontology alignment system is presented which employs an unsupervised learning method using a recursive neural network to align classes between different ontologies.

1.2 Specific techniques used

OntoConnect consists of two main tasks: the first task is unsupervised learning of the OntoConnect model with source ontology classes/concepts. The second task is the prediction of similar source classes/concepts for the corresponding target ontology class/concept using the trained OntoConnect model. Figure 1 represents a workflow of the proposed OntoConnect ontology alignment system.

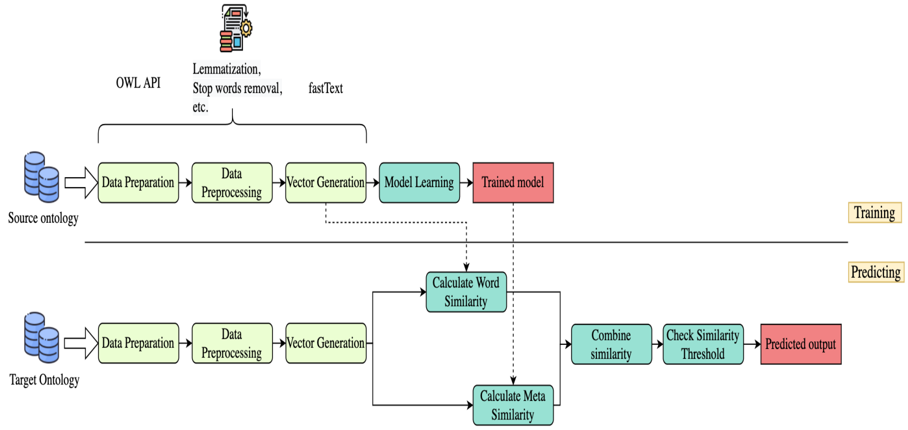

Fig. 1. Overview of OntoConnect Ontology Alignment system

(i) ***Data Preparation:*** In this step, a Java API named OWL API [8] and HermiT Reasoner [9] are used to extract meta information of a class/concept, such as IRI, label, restriction, parent, child, equivalent, and disjoint classes of each class/concept of the source ontology (S) and the target ontology (T).

(ii) ***Data Preprocessing:*** Several data preprocessing techniques are used on both source and target class/concept labels. Special characters and common stop-words in English are removed from the class/concept labels. Apart from stopword, we have used tokenization, lemmatization, conversion of roman letters to numeric, etc.

(iii) ***Vector Generation:*** In this step, a pre-trained embedding model called fastText [2] developed by Facebook's AI Research (FAIR) lab is used on the source and the target ontology class/concept to generate vectors. It treats each word as composed of character n-grams. So the vector for a word is made of the sum of this character n-grams. It helps to get a meaningful vector even when the dictionary word is not present in the model. The default dimension of the generated vector is 300.

(iv) ***Model Learning:*** Next, the vector generated for each source ontology class/concept is fed to an unsupervised recursive neural network [4]. The recursive neural network is an extension of a recurrent neural network [5]. The input to the recursive neural network is the meta-information of a source ontology class and the output is the source ontology class itself. The intuition behind this learning process is that during prediction if any target class has meta information similar to a source ontology class meta information then the model will be able to predict the same/similar vector to the source ontology class. Figure 2 shows the general architecture of the recursive neural network in OntoConnect. In the figure, $pc_1...pc_m$ denote the parent classes of a class/concept. Similarly, $cc_1...cc_n$, ec_1, dc_1, $rc'_1...rc'_s...rc''_1...rc''_t$ are child classes, equivalent class, disjoint class, and restriction classes of a class/concept. $X(pc_1)$ is the vector representation of pc_1 obtained from pre-trained fastText model. $c(pc_1)$ is the cell state and $h(pc_1)$ is the hidden state of the long short term memory (LSTM) cell [7] for parent meta information. At the output level, the model generates a vector with the same dimension as that of the input vector.

(v) ***Model Prediction:*** The word similarity is calculated by the cosine similarity between the source and target class/concept vectors. Next, the meta-information of the target ontology class is fed to the trained ontology alignment model which predicts a vector similar to one of the source classes. We use the cosine similarity to measure the meta similarity as well. A combined similarity i.e., the average of the word similarity and meta similarity, is used for the final prediction of similar class/concept.

1.3 Adaptations made for the evaluation

OntoConnect consists of two components. The first one is the java component and the second one is the python component. Figure 3 shows a high-level system architecture of OntoConnect. It follows a microservices architecture, consisting of different components that work together. The main motivation behind using microservices was to isolate different tasks and use some of the existing modules within our project. This allowed the use of different programming languages for different purposes based on their applicability. Each microservice was dockerized, making it modular, portable, as well as isolating the environments so as to run on any operating system.

We have tried to test the OntoConnect system on Semantic Evaluation At Large Scale (SEALS) [11] platform, however, were not able to run the system as SEALS only provides a wrapper for java-specific tools only. Other frameworks

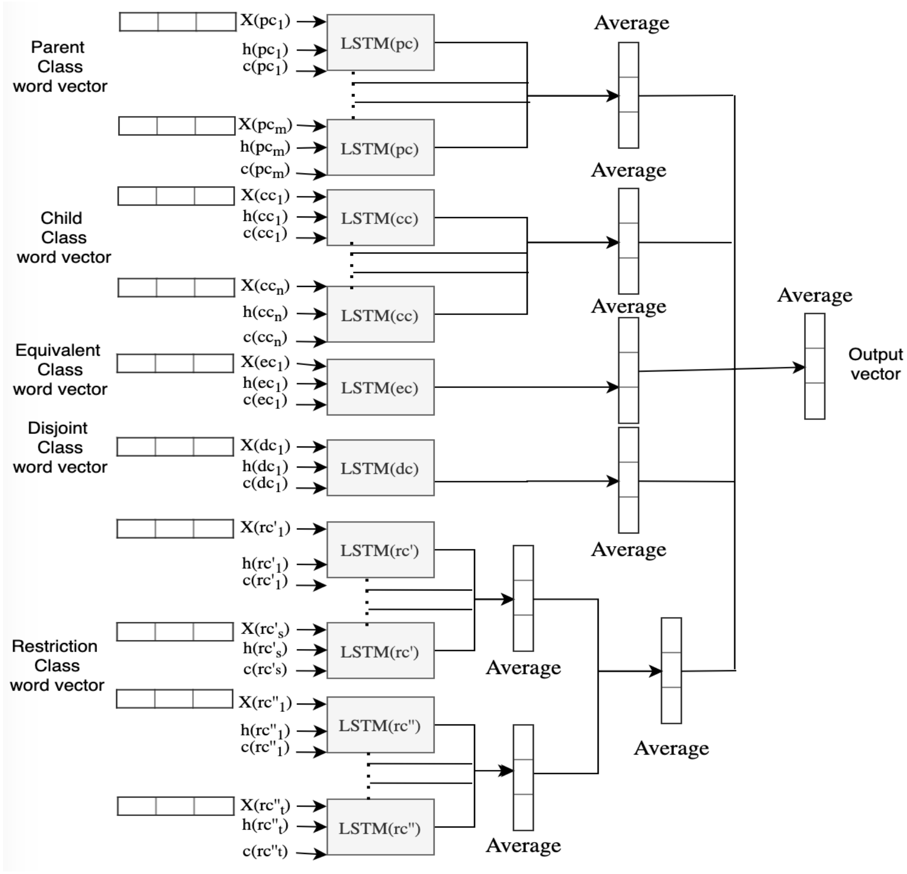

Fig. 2. Recursive Neural Network (dynamic array tree-LSTM model) of Ontology Alignment System

such as MELT [6] was also tried for the evaluation of the OntoConnect System, however, MELT provides an evaluation wrapper for either java-only tools or python-only tools. It does not support tools that have both java and python components in one. OntoConnect system uses both the java and python components. Hobbit platform [10] permits dockerized tool which is independent of the type of the programming language of the tool. For this reason, the dockerized approach is used to build the OntoConnect System and we could successfully test and evaluate it on the Hobbit platform.

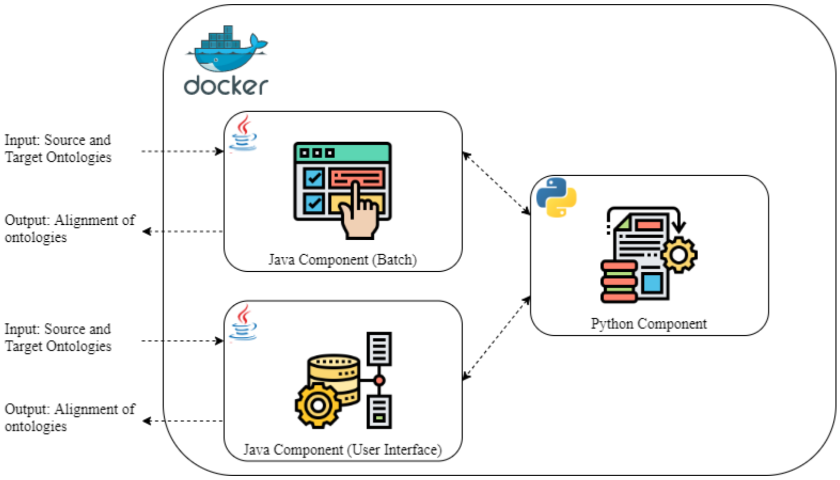
docker
Input: Source and Target Ontologies
Output: Alignment of ontologies
Java Component (Batch)
Input: Source and Target Ontologies
Output: Alignment of ontologies
Java Component (User Interface)
Python Component

Fig. 3. OntoConnect system architecture

1.4 Link to the system and parameters file

The OntoConnect code is available on GitHub: `https://github.com/dbpedia/linking`

1.5 Link to the set of provided alignments

The OntoConnect result is published on `http://oaei.ontologymatching.org/2020/results/anatomy/index.html` . The result is also available on GitHub: `https://github.com/dbpedia/linking/wiki/Result`

2 Results

We have tested OntoConnect on the Anatomy [1] data set published by OAEI with different parameters such as input vector dimension and similarity threshold. Three different files are provided in the OAEI System: source ontology, target ontology, and result or alignment file. Standard evaluation metrics, i.e., precision, recall, and F-measure are used. The OntoConnect system yields satisfactory results with a precision of 99.6%, recall of 66.5%, and F-measure of 79.7% for a similarity threshold of 0.99 with the 100-dimension input vector. Table 1 gives a summary of the result of OntoConnect on the Anatomy data set.

Table 1. OntoConnect performance in the Anatomy track

Matcher	Runtime	Precision	Recall	Recall++	F-Measure
OntoConnect	248	0.996	0.665	0.136	0.797

3 General comments

The main goal of the OntoConnect is to address questions such as, (i) can ontology alignment be done independently of domain information? (ii) Can ontology alignment be achieved by using only the meta-information and structural information of ontologies? (iii) Can ontology alignment be achieved using unsupervised machine learning instead of the traditional rule-based approaches? The OntoConnect tool is able to address all the above questions and moreover, it performs well compared to some of the state-of-the-art systems in OAEI 2020. The main strength of the tool is that a domain-independent approach is performed by achieving the mentioned goals.

Besides the strengths of the tool, there is a number of potential improvements to be realized for OntoConnect. The main weakness of the OntoConnect tool is the complex architecture of the system, as it has two different components of different languages i.e. java and python. It was difficult to incorporate any OAEI evaluation wrapper because of the complex architecture of the tool. We have used Docker to execute the system on the HOBBIT platform but there is still room for improving the system architecture so that the tool can be easily executed. The second problem is the size of the project. We have used the pre-trained model fastText in the system and the default dimension of the fastText output vector is 300. The high dimension of the vector causes an increase in the size of the tool. In future work, we would like to explore different procedures such as autoencoder approach to reduce the dimension to minimize the size of the tool.

4 Conclusion

In this study, OntoConnect tool is presented with a generic and domain-independent approach to align multiple ontologies that eliminate cumbersome and error-prone manual work. A non-linear neural network is used for feature extraction from the source ontology and is independent of the domain knowledge. Participating in this campaign for the first time allowed us to see how the OntoConnect system was performing compared to the other tools. It was seen that our tool had a high precision among the tools without any domain knowledge and without depending on any vocabularies. But both recall and F1 have room to improve. Even though OntoConnect has a reasonable runtime, we would like to decrease the execution time for better performance. We have seen that our tool is comparable to the current state-of-the-art domain-specific approaches and we would like to participate in other tracks next year to see the results in different domains.

Acknowledgement The authors gratefully acknowledge the Google Summer Code program and DBpedia organization for guidance and support. We also thank the Google Cloud Platform (GCP) research credits program for providing an environment to run the experiments using their Cloud Computing services.

Beyza Yaman has been supported by the European Union's Horizon 2020 research and innovation programme under Marie Sklodowska-Curie grant agreement No. 801522, by Science Foundation Ireland and co-funded by the European Regional Development Fund through the ADAPT Centre for Digital Content Technology [grant number 13/RC/2106] and Ordnance Survey Ireland.

References

1. http://oaei.ontologymatching.org/2020/anatomy/index.html
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
3. Chakraborty, J., Bansal, S., Yaman, B., Virgili, L., Konar, K.: Ontoconnect: Unsupervised ontology alignment with recursive neural network. In: Proceedings of the 36th ACM/SIGAPP Symposium on Applied Computing, SAC 2021, Gwangjiu, South Korea,March 22-26, 2021 (In Press)
4. Chinea, A.: Understanding the principles of recursive neural networks: a generative approach to tackle model complexity. In: International Conference on Artificial Neural Networks. pp. 952–963. Springer (2009)
5. Goller, C., Kuchler, A.: Learning task-dependent distributed representations by backpropagation through structure. In: Proceedings of International Conference on Neural Networks (ICNN'96). vol. 1, pp. 347–352. IEEE (1996)
6. Hertling, S., Portisch, J., Paulheim, H.: Melt-matching evaluation toolkit. In: International Conference on Semantic Systems. pp. 231–245. Springer, Cham (2019)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
8. Horridge, M., Bechhofer, S.: The owl api: A java api for owl ontologies. Semantic web **2**(1), 11–21 (2011)
9. Motik, B., Shearer, R., Horrocks, I.: Optimized reasoning in description logics using hypertableaux. In: International Conference on Automated Deduction. pp. 67–83. Springer (2007)
10. Röder, M., Kuchelev, D., Ngonga Ngomo, A.C.: Hobbit: A platform for benchmarking big linked data. Data Science (Preprint), 1–21 (2019)
11. Wrigley, S.N., García-Castro, R., Nixon, L.: Semantic evaluation at large scale (seals). In: Proceedings of the 21st International Conference on World Wide Web. pp. 299–302 (2012)