OntoConnect: Unsupervised Ontology Alignment with Recursive Neural Network

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

1 Introduction

Ontology alignment is a process to integrate multiple knowledge bases. Most of the current state-of-the-art ontology alignment systems depend on domain knowledge that makes the alignment process domain-specific. To address this challenge, we aim at developing an ontology alignment approach that is independent of domain knowledge. In this paper, we contribute to the OAEI initiative, an ontology alignment approach called OntoConnect is presented that exploits an unsupervised learning method using a recursive neural network to align classes between different ontologies.

2 OntoConnect System

OntoConnect consists of two main tasks: the first task is unsupervised learning of the model and the second task is the prediction of similar classes/concepts using the trained model. Figure 1a represents a workflow of the proposed OntoConnect ontology alignment system. In the ***Data Preparation*** step, a Java API named OWL API [3] is used to extract meta information of a class, 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). For ***Data Preprocessing***, the following techniques are applied on the class/concept labels such as Normalization, Stemming/Lemmatization, Stop-word removal, Tokenization, etc. In the ***Vector Generation*** 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. Next for ***Model Learning***, the vector generated for each source ontology class/concept is fed to a unsupervised recursive neural network. 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's meta information then the model will be able to predict

the same/similar vector to the source ontology class. Figure 1b 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 LSTM cell for parent meta information. At the output level, the model generates a vector with the same dimension as that of the input vector. In ***Model Prediction***, first, the word similarity is calculated by the cosine distance 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 the similar class/concept.

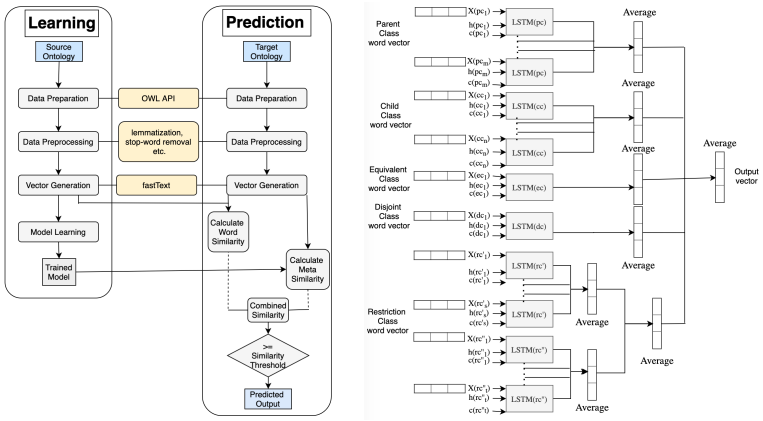

Fig. 1: (a) Overview of OntoConnect System (b) Recursive Neural Network of OntoConnect System

3 Experiments

For the Anatomy data set published by OAEI [1], 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.

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. Horridge, M., Bechhofer, S.: The owl api: A java api for owl ontologies. Semantic web **2**(1), 11–21 (2011)