Preprint/review version to be published in **ICMLA21**:
Proceedings of the IEEE International Conference on Machine Learning and Applications, 2021
Jaydeep Chakraborty, Mohamed Ahmed Sherif, Hamada M. Zahera, Srividya Bansal

OntoConnect: Domain-Agnostic Ontology Alignment using Graph Embedding with Negative Sampling

Anonymous
for
review

Abstract—The ontology alignment task aims at linking two or more different ontologies from the same domain or different domains. Over the years, many techniques have been proposed for ontology instance alignment, schema alignment, and link discovery. Most of the available approaches require human intervention or work within a specif c domain and follow a rule-based and logic-based approach. In this paper, we present an ontology alignment approach using graph embedding with negative sampling that is independent of the domain and does not require any human intervention. The graph neural network with negative sampling enriches the embedding of a class/concept by encapsulating the ontological structure of the class/concept. The experimental results show that our proposed unsupervised learning-based approach successfully captures the structural information or meta-information of ontological classes and predicts the correct correspondence list with accuracy comparable to other state-of-the-art systems.

Index Terms—Learning-based Alignment, Graph Neural Network, Graph embedding with negative sampling, Ontology Schema Alignment

I. Introduction

An ontology is a vocabulary that provides a formal description of entities within a domain and their relationships with other entities. Along with basic schema information, ontology also captures information in the form of metadata about cardinality, restrictions, hierarchy and semantic meaning. With the rapid growth of semantic (RDF) data on the web, many organizations like DBpedia, Earth Science Information Partners (ESIP) are publishing more and more data in RDF format. The ontology alignment task aims at linking two or more different ontologies from the same domain or different domains. In particular, it is the process of f nding the semantic relationships between two or more ontological entities and/or instances. Information/data sharing among different systems is quite limited because of differences in data based on syntax, structures, and semantics. Ontology alignment is used to overcome the limitation of semantic interoperability of current vast distributed systems available on the Web.

In spite of the availability of large hierarchical domain-specif c datasets, automated ontology mapping is still a complex problem. Over the years, many techniques have been proposed for ontology instance alignment, schema alignment, and link discovery. Most of the available approaches such as COMA++ [1], GLUE [2] require human intervention or work within a specif c domain. The challenge involves representing an entity as a vector that encodes all context information of the entity such as hierarchical information, properties, constraints, etc. The ontological representation is rich in comparison with the regular data schema because of metadata about various properties, constraints, domain membership and relationship to other entities within the domain. While f nding similarities between entities this metadata is often overlooked. The second challenge is that the comparison of two ontologies is an intense operation and highly depends on the domain and the language that the ontologies are expressed in. Most current methods require human intervention that leads to a time-consuming and cumbersome process with output being human error-prone.

To address this challenge, we aim at developing an unsupervised ontology alignment approach that is independent of domain knowledge and does not need domain expert intervention. Our proposed approach explores the use of machine learning techniques in ontology linking in an unsupervised manner with no or minimum background knowledge and compares the results with state-of-the-art tools. Our goal is to perform ontology mapping without any domain expert intervention using an unsupervised neural approach utilizing metadata of ontological entities and evaluate the performance of the system in comparison with state-of-the-art systems.

The work presented in this paper addresses the following research questions.
(i) Can ontology alignment be achieved using unsupervised machine learning with graph neural networks instead of the traditional rule-based approaches?
(ii) Can ontology alignment be done independent of domain information and without the need for domain expert intervention?
(iii) Can ontology alignment be improved using the meta-information and structural information of ontologies?

To the best of our knowledge, our study is the f rst to present a domain-agnostic unsupervised learning approach to deal with the ontology alignment. Our experiments using our proposed graph embedding with negative sampling achieves performance similar to current state-of-the art systems with the added advantage of being domain-independent and not requiring human intervention.

II. Related Work

Traditional data integration approaches are mainly applicable to relational data models. There are mainly two approaches used in traditional schema mapping. The first approach is called global-as-view which requires that the global schema must be expressed in terms of the data sources. The second approach is called local-as-view which requires the global schema has to be specified independently from the sources, and the relationships between the global schema and the sources are established by defining every source as a view over the global schema [3], [4], [5]. As relational data models do not have semantic information, the traditional data integration techniques often fail for ontology alignment.

The matching process for ontologies is different compared to traditional data as an ontology captures semantic information in the form hierarchical relationships. There has been lot of research in the area of ontology matching. Ontology Alignment Evaluation Initiative, i.e., OAEI [6] is one such initiative. It is a coordinated international initiative and an annual competition that organizes the evaluation of different ontology matching systems and provides benchmark evaluation. There are a number of different proposed approaches available with good results. However, these approaches have their advantages and disadvantages.

Non-Learning based methods: There are number of techniques such as syntactic-based, structural-based, semantic-based, extensional-based that have been explored over time. For example, tools like LIMES [7], AGREEMENTMAKER LIGHT [8], and COMA++ [1] use different type of distance metrics such as Hamming, Levenshtein and Jaro-Winkler distances for the ontology alignment process. In structural-based approach, external linguistic resources like WordNet [9] (a machine-readable dictionary) is used for finding similarity between entities. Different distance metrics such as *wu-palmer* metric [10], *resnik* similarity [11], *lin* similarity [12], *jiang-conrath* distance [13] are used to calculate the path similarity between entities. On the other hand, there are semantic-based approaches like SAT, description logic, rule-based inference, Propositional Horn satisfiability. Most of the state-of-the-art tools such as S-Match [14], CTXMATCH/CTXMATCH2 [15], BLOOMS/BLOOMS+ [16], LogMap/LogMap2 [17], Paris [18] use rule-based or logic-based approach extensively. Some of these approaches use an extensional-based approach, i.e., use instance or individual representations to find similarity between two classes/concepts. GLUE [2], RiMOM [19], and ObjectCoref [20] are such tools using extensional-based approaches for ontology alignment. These approaches are customized based on knowledge of a particular domain with the use of specialized vocabularies or a domain-expert and cannot be used across multiple domains.

Learning-based methods: Besides traditional and rule-based approaches, nowadays many are exploring learning-based approaches in ontology alignment. Tools like Context and Inference-based alignER (CIDER) [21], GLUE [2], Yet Another Matcher (YAM++) [22], DL-Learner [23] systems use heuristic learning methods. On the other hand, approaches like Zhang et al. [24], OntoEmma [25], DeepFCA [26], and VeeAlign [27] use pre-trained embedding for ontology entities in neural network. On the other hand, Chen et al. [28] uses a semi-supervised approach to the traditional base tool LogMap to enhance the ontology alignment process. Wang et al. [29] propose a graph convolution network to embed entities from different languages into a unified vector space, where equivalent entities are expected to be as close as possible. These approaches use semi-supervised approaches with labeling done with the help of a domain expert. In contrast, the approach presented in this paper is a domain-agnostic and completely unsupervised approach that takes advantage of the hierarchical nature of the meta-data about the various concepts described in the ontology through graph embedding and presents performance results in comparison with other existing systems.

TABLE I: Challenges with Ontology Alignment

	Mouse Ontology	**Human Ontology**
Case-A *(partially different)*	lunate	Lunate_Bone
	spleen periarteriolar lymphatic sheath	Periarteriolar_Lymphoid _Sheath
	lumbar vertebra 3	L3_Vertebra
	trochlear IV nerve	Trochlear_Nerve
	hair shaft	Shaft_of_the_Hair
Case-B *(completely different)*	cranium	Skull
	external naris	Nostril

III. Approach

The lack of ground truth is a massive problem in the evaluation of different ontology alignment systems. The differences in literal expressions of the ontological entities make the ontology alignment unpredictable. String-based approaches fail to calculate correct similarity in ontology alignment. Entity names that are syntactically similar could have different naming conventions used, e.g., “E-mail” vs. “email”, “url” vs “U.R.L.”. Entities could be synonyms. e.g., “Participant” vs. “Attendee” or similar in meaning in a specific domain, e.g., “contribution” vs. “paper” in the conference organization domain. The representation could be in abbreviated forms. e.g., “PC Member” vs.“ProgramCommitteeMember”, or acronym, e.g., “WWW” vs “World Wide Web”. Entity names that are tokenizable and their tokens (or only a part of tokens) are syntactic or meaning similar. e.g., “hasSurname” vs. “has the last name”, “Camera-ready contribution” vs “Final manuscript”. They could be partially syntactically similar. e.g., “email” vs “hasEmail”, or “Regular author” vs. “author”.

Table I shows different challenges in the ontology alignment process with the examples from the Anatomy dataset [30]. Case-A contains human and mouse anatomy classes/concepts which are partially different. For example, "lunate" from mouse ontology and "Lunate_Bone" from the human ontology are presenting the same body part but their label is partially different syntactically. Another example is "lumbar vertebra 3" and "L3_Vertebra". In this example "L" is an abbreviation of "lumbar". For the Case-B, the ontological classes/concepts are completely different syntactically. Here "cranium" and "skull" are semantically synonymous but syntactically completely different.

Our proposed approach is a combination of syntactic and structural learning-based methods that exploits the meta-information of the ontological classes (hierarchy, parent-child classes, restrictions, equivalent class, and disjoint class). We use a pre-trained embedding model to generate a meaningful vector for all ontological concepts. These vectors can then be fed into a graph neural network to train a model and use it for prediction.

A. Graph Embedding

We deployed a graph embedding approach that transforms nodes and edges into a vector space that encapsulates the structure of a graph and information about neighboring nodes. Our goal is compare an entity in a given source ontology to entities in a given target ontology in order to identify entities that are similar. In an unsupervised learning-based approach, the goal is to use the structure of the source ontology and come up with an embedding that is later compared for similarity with entities of a destination ontology. In our study, three different graph neural network methods are explored. The first graph neural method is the *Graph Convolution Network* (GCN) that uses general neural learning concepts to graph like data and enhance the embedding of each node by a message passing scheme. In the paper [31], the authors describe a scalable approach for semi-supervised learning on graph-structured data that is based on an efficient variant of convolutional neural networks which operate directly on graphs. The GCN layer defined in [31] is described in equation 1, where $\mathbf{x}_v^{(\ell+1)}$ is the node features of all nodes $v \in \mathcal{V}$ in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathbf{W}^{(\ell+1)}$ denotes a trainable weight matrix and $c_{w,v}$ refers to a fixed normalization coefficient for each edge. In GCN, the weights of edges are defined explicitly.

$$\mathbf{x}_v^{(\ell+1)} = \mathbf{W}^{(\ell+1)} \sum_{w \in \mathcal{N}(v) \,\cup\, \{v\}} \frac{1}{c_{w,v}} \cdot \mathbf{x}_w^{(\ell)} \tag{1}$$

The second graph neural method explored is the *Graph Attention Network* (GAT) [32] that determines the weights of edges of the graph implicitly. GAT employs self attention over the node features, where the embeddings from the neighbors are aggregated together and scaled by the attention layer. Every neighbor node of a node sends its own embedding of attentional coefficients. The attentional coefficients per each attention head are used to compute a separate linear combination of neighbours' features or embedding. GAT aggregates all the combinations by concatenation or averaging to obtain the next level features of the node. *Both GCN and GAT work well in the transductive and inductive approaches that involve observing specific labeled training cases. The challenge we are addressing is ontology alignment in the absence of labeled data and therefore the learning has to be completely unsupervised.*

As an alternative, the third approach is *Graph embedding with negative sampling* is explored. PyTorch-BigGraph (PBG) [33] is a distributed system for learning graph embeddings for large graphs. PBG operates on graphs with vertices having multiple edges. Here the vertices are called entities and the edges are the relation between source and destination entity. A multi-relation graph is a directed graph $G = (V, R, E)$ where V are the nodes or entities, R is a set of relations, and E is a set of edges. In the graph, a generic element $e = (s, r, d)$ where s is the source node/entity, d is the destination node/entity, and r is the relation between s and d. Here, $s, d \in V$ and $r \in R$. Equation 2 is the scoring function used in PBG, where θ_s is the source entity, θ_d is the destination entity, and θ_r is the relation between them. $g_s(\theta_s, \theta_r)$ is the "complex-diagonal" operator between θ_s and θ_r. On the other hand, $g_d(\theta_d, \theta_r)$ is the "complex-diagonal" operator between θ_d and θ_r. PBG tries to maximize the scoring function $f(\theta_s, \theta_r, \theta_d)$ for any $(s, r, d) \in E$ and minimizes it for $(s, r, d) \notin E$.

$$f(\theta_s, \theta_r, \theta_d) = sim(g_s(\theta_s, \theta_r), g_d(\theta_d, \theta_r)) \tag{2}$$

The learning principle of the PyTorch-BigGraph (PBG) is to find embeddings for the entities so the distance between the neighbor nodes and the entity should be closer. On the other hand, the distance between non-neighbor nodes and the entity should be longer. The edges of the input graph data given to the PBG model are treated as positive edges. It produces a set of negative edges for each positive edge. It produces negative samples for a given positive edge by a corrupted version of the entity on one side and keeping the other side intact [34]. PBG uses different ways to sample negative edges. For our experiment, "all negative" is used to generate the negative samples. "all negative" sampling method creates a negative edge/relation between the source and the destination node. Let's say r is a positive edge between two entities s and d. For each such positive edge (s, r, d) there will be a negative edge (s', r, d) between s' and d where s' is of the same entity type as s. Besides this, there will be another negative edge (s, r, d') between s and d' where d' is of the same entity type as d.

Equation 3 is the loss ($\mathcal{L}$) function used in PBG. The main idea of PBG is to maximize the scores of positive edges and minimize the scores of negative edges. Here, G is a list of edges. S'_e is the set of negative edges for every positive edge. $f(e)$ is the score of a positive edge and $f(e')$ is the score of a negative edge. λ is the regularization parameter.

$$\begin{gathered} \mathcal{L} = \sum_{e \in G} \sum_{e' \in S'_e} max(f(e) - f(e') + \lambda, 0) \\ S'_e = (s', r, d) | s' \in V \cup (s, r, d') | d' \in V \end{gathered} \tag{3}$$

PBG updates the embeddings and related parameters in minibatch stochastic gradient descent (SGD). It uses an Ada-

Algorithm 1: *OntoConnect* Graph Embedding algorithm

```
Input: Set of concepts in source and target ontology
Output: Graph Embedding of source and target ontology $concepts$
1 $onto\_graph\_embed\_set \leftarrow \emptyset$
2 foreach $concept\ c_i \in concepts$ do
      /* Retrive vector from pre-trained model FastText for the concept $c_i$ */
3     $Vec_{c_i} \leftarrow getVector(c_i)$
      /* Retrive meta-information (parent, child, equivalent, disjoint,
         restriction) of the concept $c_i$ from the ontology. */
4     $Meta_{c_i} \leftarrow getMetaInformation(c_i)$
      /* Retrive vector from pre-trained model FastText for each
         meta-information of the concept $c_i$ */
5     $MetaVec_{c_i} \leftarrow getVector(Meta_{c_i})$
      /* Retrieve Embedding for the concept $c_i$ stochastically */
6     $onto\_graph\_embed\_set \leftarrow getEmbed(Vec_{c_i}, MetaVec_{c_i})$
7 return $onto\_graph\_embed\_set$
```

grad optimizer and sums the accumulated gradient over each embedding vector to reduce the memory usage on large graphs [35].

The main motivation of graph embedding with negative sampling is generating embedding of nodes that uses its neighbor nodes' information. This approach does not need any labeled data for the training and we can fit our unlabeled graph-structured data easily. In the following section, the graph embedding with negative sampling approach is discussed in detail.

B. *Graph Embedding with negative sampling Approach*

In this section we present an Ontology alignment system called ONTOCONNECT that uses graph embedding with negative sampling. The overall ONTOCONNECT alignment process with graph neural network can be divided into two main processes. The first one is generating the embedding of the source ontology classes/concepts and the target ontology classes/concepts. The second one is calculating the similarity score between source classes/concepts for each target class/concept to form correspondences.

The strategy is to train a graph neural network model in a stochastic manner with each meta-information of source and target classes/concepts. The generation of embedding of the source and target ontology classes/concepts is described in Algorithm 1 shown below. The first step in the algorithm is to extract the meta-information for each source class/concept. The function "getMetaInformation" is responsible for collecting the meta-information for a class from the ontology. The function "getVector" retrieves the word-vector for each source class/concept from a pre-trained model. The function "getEmbed" trains the graph neural model for each class/concept and encodes each class/concept into an embedding which captures the meta information or structural information of that particular class/concept.

Algorithm 2 shows the algorithm for calculating the similarity between the source and target ontology classes/concepts. The method "getVector" retrieves the word-vector for a class/concept for both source and target ontology. Now, for a target class/concept, we are calculating similarity for each source ontology class/concept. The word similarity ($Word_{sim_{i,j}}$) is

Algorithm 2: *OntoConnect* Similarity Calculation algorithm

```
Input: set of concepts in source ontology $sConcepts$, set of concepts in
       target ontology $tConcepts$
Output: List of Correspondences $correspondences$
1 $correspondences \leftarrow \emptyset$
2 foreach $target\ concept\ t_i \in tConcepts$ do
      /* Retrive vector from pre-trained model FastText for the target
         concept $t_i$ */
3     $Vec_{t_i} \leftarrow getVector(t_i)$
      /* Retrive graph embedding of the target concept $t_i$ */
4     $Embed_{t_i} \leftarrow getEmbedVec(t_i)$
5     $Cmb_{sim_i} \leftarrow \emptyset$
6     $correspondence_i \leftarrow \emptyset$
7     foreach $source\ concept\ s_j \in sConcepts$ do
          /* Retrive vector from pre-trained model FastText for the source
             concept $s_j$ */
8         $Vec_{s_j} \leftarrow getVector(s_j)$
          /* Compute cosine-similarity between $Vec_{t_i}, Vec_{s_j}$ */
9         $Word_{sim_{i,j}} \leftarrow computeSimilarity(Vec_{t_i}, Vec_{s_j})$
          /* Retrive graph embedding of the source concept $s_j$ */
10        $Embed_{s_j} \leftarrow getEmbedVec(s_j)$
          /* Compute cosine-similarity between $Embed_{t_i}, Embed_{s_j}$ */
11        $Meta_{sim_{i,j}} \leftarrow computeSimilarity(Embed_{t_i}, Embed_{s_j})$
12        $TmpCmb_{sim_{i,j}} \leftarrow$
            $weightedHarmonicMean(Word_{sim_{i,j}}, Meta_{sim_{i,j}})$
          /* Updating $correspondence_i$ based on combined similarity */
13        if $TmpCmb_{sim_{i,j}} > Cmb_{sim_i}$ then
14            $Cmb_{sim_i} \leftarrow TmpCmb_{sim_{i,j}}$
              $correspondence_i \leftarrow [t_i, s_j, Cmb_{sim_i}]$
      /* Updating $correspondences$ list */
15    $correspondences \leftarrow correspondences + correspondence_i$
16 return $correspondences$
```

the cosine similarity between the target and source ontology class/concept word-vectors retrieved from a pre-trained model (described in the next section). On the other hand, meta-similarity ($Meta_{sim_{i,j}}$) is the cosine similarity between the target and source ontology class/concept graph-embedding from the graph neural network model. Next, a weighted combined similarity of the word-similarity ($Word_{sim_{i,j}}$) and meta-similarity ($Meta_{sim_{i,j}}$) is calculated for each pair of the target and source ontology classes/concepts. We are considering the source ontology class/concept as a similar class/concept with the maximum combined similarity (Cmb_{sim_i}). The correspondence list is updated with the most similar class/concept pair having the highest similarity score. In the end, it returns the correspondence list.

C. *Pre-trained Word embedding model*

We explored and experimented with a number of different word embedding models to identify the right fit for ONTOCONNECT. The first word embedding technique is Word2vec [36] that belongs to a family of model architectures and optimizations that used to learn word embeddings from large datasets. In Word2vec, there are two novel model architectures that compute continuous vector representations of words from a very large corpus. The two model architectures are the continuous bag-of-words model (CBOW) and the continuous skip-gram model. In the continuous bag-of-words model, the current word is predicted from a window of surrounding context words on the other hand, in the continuous skip-gram model, the model predicts the surrounding window of context words for the current word. Apart from Word2vec, GloVe [37] is another technique to obtain the word embedding. GloVe or Global Vectors for Word Representation uses training on aggregated global word-word co-occurrence statistics from a corpus. It encodes the co-occurrence probability ratio between two words. Both Word2vec and GloVe don't work well for corpus with new or unseen words. Our approach used unsupervised learning that is domain-agnostic and therefore needs the ability to handle unseen words.

In our experiments, we have also explored BERT or Bidirectional Encoder Representations from Transformer [38]. BERT is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weights between them are dynamically calculated based upon their connection. It is designed to read or encode sentences from both directions left-to-right and right-to-left. The main difference between BERT and Word2vec is BERT can produce different word representations for the same word in different sentences. For example, given two sentences: "The man was accused of robbing a bank." and "The man went fishing by the bank of the river." In both sentences, Word2vec produces the same word embedding for the word "bank" while BERT generates different embeddings as the context is different for the word "bank" in those two sentences. This model is suitable for corpus with sentences where the position of the word in a sentence and its context in comparison with other words in the sentence is important. However, for the ontology alignment problem we are comparing ontological terms that are typically words or phrases and not sentences.

Another pre-trained embedding models that is currently popular is FastText [39] developed by Facebook's AI Research (FAIR) lab. This embedding model uses the skip-gram technique, where each word is considered as a bag of n-gram characters. It learns the representations for character n-grams and represents the words as the sum of the n-gram vectors. The main advantage of the model provided by FastText is that it can generate a meaningful vector for a word that is not present in its dictionary. In our approach, we used FastText model trained on Wikipedia 2017, UMBC WebBase corpus, and statmt.org news data set, which generates a 300-dimension vector for each word. The main reason behind using FastText within OntoConnect is that it can generate meaningful vectors of the ontology class/concept even when the word is not present in the model.

D. OntoConnect *System Workflow*

Our proposed ontology alignment system, OntoConnect, consists of two main tasks: (1) Generating the embedding for each class/concept in the source and target ontology, and (2) Calculating the similarity score between the source and target ontology class from the embedding and generate correspondence list. Figure 1 represents a workflow of the proposed OntoConnect system using Graph Neural Network.

In the Graph Neural Network, in the first step, OWL API is used to extract the meta-information of each class from the

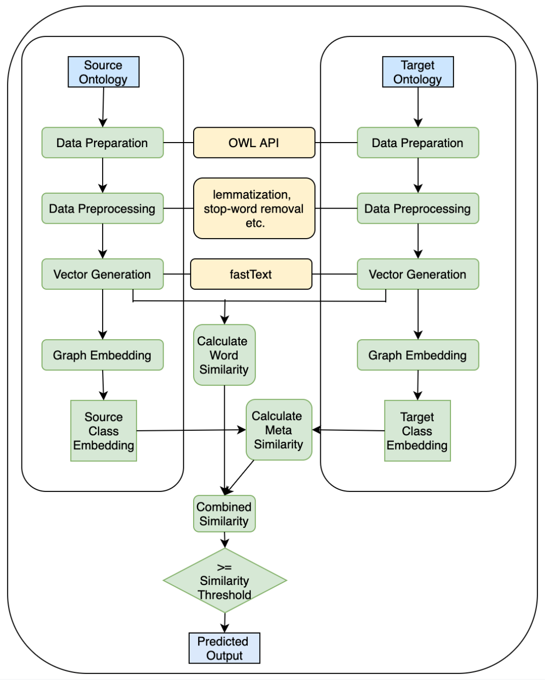


Fig. 1: Project Flow of OntoConnect - Graph Neural Network

source ontology and target ontology. In the next step, data preprocessing techniques such as Lemmatisation and stop-word removal are used on the labels of the extracted class. After that, we have used a pre-trained model FastText [39] to generate vectors from each source and target ontological classes. Each class of the source and target and its meta-information are fed to the Graph Neural network which produces entity embedding for each entity. In the next phase, both word-similarity and meta-similarity are calculated for each pair of classes of source and target ontologies. Based on the similarity score, the list of correspondence is generated.

E. Graph Entity Embedding

In this step, the context of classes is used to make its vector representation semantically richer. From the ontology matching model, a class can have attributes like Parent Classes, Child Classes, Equivalent Classes, Disjoint Classes, and Restrictions. Figure 2 shows an example of training data which will be used in training the Graph Neural Network. Assume the class is "c_1". "c_1" has meta information "parent class", "child class", "equivalent class", "disjoint class" and "restriction class". The parent class of "c_1" is "pc_1c_1". "cc_1c_1", "cc_2c_1", "ec_1c_1", "dc_1c_1", "$rc_1'c_1$", "$rc_1''c_1$" are child classes, equivalent class, disjoint class and restriction classes of "c_1" respectively. In this process, the class of source and target ontology are encoded into vector or embedding which incorporates its meta-information.

F. Correspondence Generation

Given two ontologies where O is the source ontology and O' is the target ontology, an alignment between these two

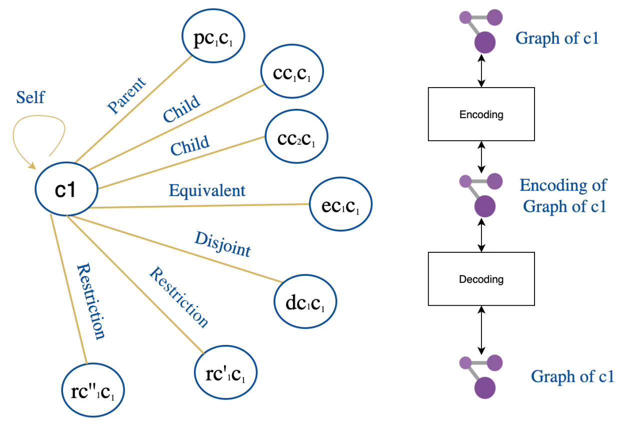


Fig. 2: Graph structure of a sample ontology class and High level representation of Graph Embedding process

ontologies is a set of correspondences $< e, e', r, n >$. Here, $e \subseteq O$ is an entity from source ontology, and $e \subseteq O'$ is an entity from target ontology. r is the relationship between e and e'. In our project relationship will be equivalence ,i.e., $=$.

To generate the correspondence, two similarity values are calculated. The first one is word similarity ($Word_{sim}$) and the second one is the meta similarity ($Meta_{sim}$). In our experiment, we have calculated a combined similarity (Cmb_{sim}) by the harmonic mean of the word and meta similarity.

The word similarity ($Word_{sim}$) is the cosine distance between the source and target class/concept vectors. Equation 4 shows the cosine similarity measurement between two vectors $\overrightarrow{s}$ and $\overrightarrow{t}$, which is the cosine of the angle projected in a multi-dimensional (d) space. In our experiment, the time complexity of this step is $O(mn)$, where m is the number of source ontology classes and n is the number of target ontology classes.

$$\cos(\overrightarrow{s}, \overrightarrow{t}) = \frac{\overrightarrow{s} \cdot \overrightarrow{t}}{\|\overrightarrow{s}\|\|\overrightarrow{t}\|} = \frac{\sum_{i=1}^{d} \mathbf{s}_i \mathbf{t}_i}{\sqrt{\sum_{i=1}^{d} (\mathbf{s}_i)^2}\sqrt{\sum_{i=1}^{d} (\mathbf{t}_i)^2}} \quad (4)$$

The meta similarity ($Meta_{sim}$) is the cosine distance between the source and target class/concept graph embedding. Equation 5 shows the cosine similarity measurement between two vectors $\overrightarrow{s_g}$ and $\overrightarrow{t_g}$, which is the cosine of the angle projected in a multi-dimensional (d) space. In our experiment, the time complexity of this step is $O(mn)$, where m is the number of source ontology classes and n is the number of target ontology classes.

$$\cos(\overrightarrow{s_g}, \overrightarrow{t_g}) = \frac{\overrightarrow{s_g} \cdot \overrightarrow{t_g}}{\|\overrightarrow{s_g}\|\|\overrightarrow{t_g}\|} = \frac{\sum_{i=1}^{d} \mathbf{s_g}_i \mathbf{t_g}_i}{\sqrt{\sum_{i=1}^{d} (\mathbf{s_g}_i)^2}\sqrt{\sum_{i=1}^{d} (\mathbf{t_g}_i)^2}} \quad (5)$$

IV. Experimental Study

Evaluation Methodology & Metrics: For evaluation, we have conducted an intrinsic evaluation that automatically compares computed similar concepts with gold standard provided by the Ontology Alignment and Evaluation Initiative (OAEI). In order to evaluate the performance of the ontology matching system, we have used standard precision and recall measurement [40]. Given two ontologies where O is the source ontology and O' is the target ontology, an alignment between these two ontologies is a set of correspondences, i.e., $< e, e', r, n >$ where, e is an entity from source ontology ($e \subseteq O$) and e' is an entity from target ontology ($e \subseteq O'$). r is the relationship between e and e'. In our work, relationship will be equivalence, i.e., $=$. n is the similarity value or confidence value [0..1] of the relationship r between e and e'. The output alignment of the ontology matching process is denoted by A. The gold copy of the reference alignment is denoted by R. The alignment produced is evaluated in terms of precision, recall, and F-measure.

Experimental Setup: The OntoConnect system was tested on a Unix system with 16 GB RAM and 100GB of disk space. Java 1.8, Python 3.6, PyTorch v1.7, PyTorch-BigGraph (PBG) are used to develop the ontology alignment system. The source code and details of dependencies and libraries can be find at this location [1].

Results & Findings: We have tested OntoConnect(Graph Neural Network) on OAEI Anatomy [30] dataset. We have varied parameters, i.e., similarity threshold and class vector dimension. For each combination of parameters the precision, recall, and F-measure are calculated for top-k predictions of target class where k=1,3,5 with 100, 200, 300 dimension word-vector. Table II shows the outcome for each combination of parameters.

The average precision, recall and F-measure across all the class/concept vector dimensions (100, 200, 300) and the number of target class predictions (k=1,3,5) are presented in Figure 3 (a). It shows the change in precision, recall, and F-measure of the system with different similarity values. It can be observed that the precision increases whereas the recall decreases with the increase in the similarity threshold value. This is expected as a higher similarity threshold causes the system to return a lower number of correspondences compared to the number of existing correspondences in the reference alignment. *From the result analysis, it can be noted that with a 0.97 similarity threshold value, the proposed ontology alignment system using Graph embedding with negative sampling approach performs best with the highest average F-measure value of 81.0%.*

Similarly, Figure 3 (b) shows the change in the average precision, recall, and F-measure values with different lengths of class vector dimensions across all the similarity threshold values and the top-k ($k = 1, 2, 3$) predicted target classes. It can be observed that only a minuscule change in the performance of the ontology alignment system occurs with the increase of the class vector dimensions. Thus, it can be noted that the performance of the proposed system stays *almost invariant towards the change in class vector length.*

[1]https://drive.google.com/drive/folders/1RtOFbZiT9aZBa51v8G3Df4G1GZHdR3NS

TABLE II: Experimental results of the OntoConnect for varying similarity thresholds (δ): Precision (P), Recall (R), and F-measure (F) values for class/concept vector dimensions $= 100, 200, 300$ and number of predictions $k = 1, 3, 5$

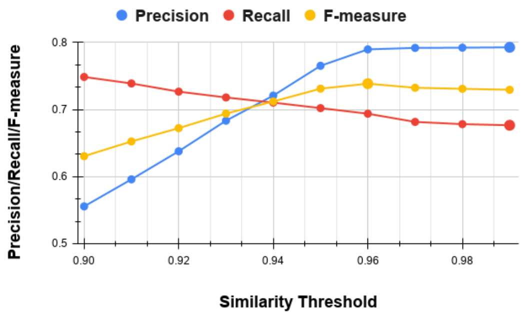
		100-dim.			**200-dim.**			**300-dim.**		
	δ	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
Top-1	0.99	97.4	67.4	79.6	97.9	67.0	79.5	97.9	66.9	79.5
	0.98	93.5	71.0	80.7	96.7	69.2	80.6	97.3	68.3	80.3
	0.97	85.3	73.0	78.7	93.0	71.5	80.8	95.3	70.5	**81.0**
	0.96	75.1	75.2	75.1	88.0	73.3	80	91.6	72.1	80.7
	0.95	67.8	76.6	71.9	80.8	75.0	77.7	86.8	73.6	79.7
	0.94	61.9	77.5	68.8	74.6	76.4	75.5	81.5	75.0	78.1
	0.93	57.8	78.6	66.6	69.2	77.3	73.0	75.4	76.4	75.9
	0.92	55.3	79.2	65.1	64.8	78.6	71.1	71.3	77.4	74.2
	0.91	53.2	80.4	64.1	61.2	79.4	69.0	67.1	78.3	72.3
	0.90	51.2	81.0	62.8	58.1	80.1	67.3	64.1	78.9	70.8
Top-3	0.99	97.6	67.6	79.9	98.1	67.1	79.7	98.2	67.1	79.7
	0.98	94.0	71.4	81.1	96.9	69.3	80.8	97.6	68.5	80.5
	0.97	86.2	73.7	79.5	93.4	71.8	81.2	95.6	70.6	81.2
	0.96	76.4	76.5	76.6	88.5	73.7	80.5	92.0	72.4	81.0
	0.95	69.7	78.7	73.9	81.5	75.7	78.4	87.3	74.0	80.2
	0.94	63.7	79.9	70.9	75.8	77.6	76.7	82.1	75.6	78.7
	0.93	59.8	81.3	68.9	70.7	79.0	74.6	76.4	77.4	76.9
	0.92	57.4	82.2	67.6	66.5	80.7	72.9	72.5	78.7	75.5
	0.91	55.4	83.6	66.7	62.8	81.6	70.9	68.7	80.2	74.0
	0.90	53.4	84.4	65.4	59.8	82.5	69.4	65.8	81.1	72.6
Top-5	0.99	97.6	67.6	79.9	98.1	67.1	79.7	98.2	67.1	79.7
	0.98	94.0	71.4	81.1	96.9	69.3	80.8	97.6	68.5	80.5
	0.97	86.3	73.8	79.6	93.4	71.8	81.2	95.5	70.7	81.2
	0.96	76.6	76.7	76.7	88.5	73.7	80.5	92.0	72.4	81.0
	0.95	69.9	79.0	74.2	81.5	75.7	78.4	87.3	74.0	80.2
	0.94	64.0	80.2	71.2	76.0	77.9	76.9	82.2	75.7	78.8
	0.93	60.3	82.0	69.5	70.9	79.3	74.9	76.6	77.6	77.1
	0.92	57.9	83	68.2	66.9	81.2	73.4	72.8	79.0	75.8
	0.91	56.2	84.8	67.6	63.2	82.1	71.4	69.0	80.5	74.3
	0.90	54.2	85.6	66.4	60.4	83.2	70.0	66.2	81.5	73.0

The average precision, recall, and F-measure of the ontology alignment system with different values of k in the top-k predicted target class across all the class vector dimensions and the similarity threshold values are presented in Figure 3 (c). The figure shows that *there is only a minuscule change in the performance of the ontology alignment system with the increase of the number of target class predictions*. Therefore, our tool may be used as an autonomous tool (i.e., without any human intervention) or an assistive tool to help a domain expert by reducing the search space for ontology alignment in any domain. In comparison with systems presented in OAEI 2020 Anatomy Challenge [6], our proposed system has results[2] comparable to other domain-specific systems in spite of not using any domain knowledge to produce alignments.

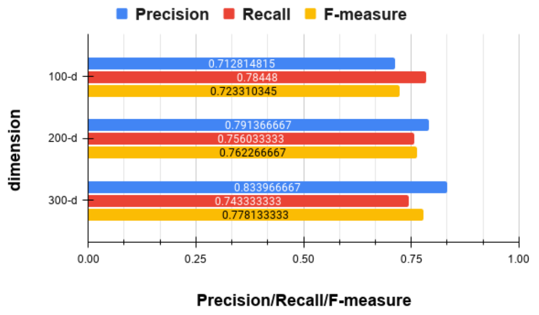
V. Conclusion

In this study, the OntoConnect system is presented that uses a generic and domain-independent approach to align multiple ontologies thereby eliminating cumbersome and error-prone manual work. A non-linear neural network (graph neural network) is used for feature extraction from the source ontology and is independent of the domain knowledge. There

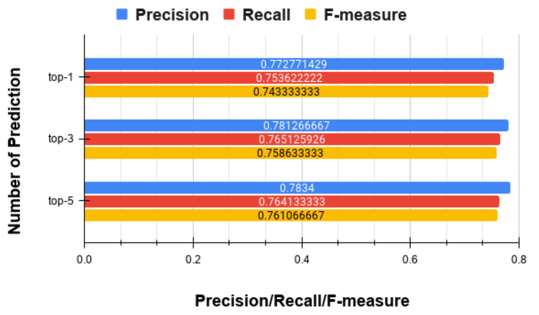
[2]http://oaei.ontologymatching.org/2020/results/anatomy/index.html



(a) Change in precision, recall, F-measure of OntoConnect with increasing similarity threshold



(b) Performance of OntoConnect with Graph Neural Network for different class/concept vector dimensions (100d, 200d, 300d)



(c) Performance of OntoConnect with Graph Neural Network for top-k predictions with k = 1, 3, 5

Fig. 3: OntoConnect Experimental results on Anatomy dataset

are numerous Neural network approaches for unsupervised machine learning. Our study shows the graph embedding with negative sampling is well suited for domain-agnostic ontology alignment in unsupervised setting. The other key findings are that a pre-trained word embedding model such as FastText is suitable for vector generation. OntoConnect system is comparable with other state-of-the-art systems (that work for domain-specific data) that competed in the OAEI 2020 challenge and has a high precision without the use of any domain-specific knowledge. In future, OntoConnect will be tested on datasets like Large Bio Medical ontology [41] that has large number of instances and a complex ontological structure. It will be evaluated in a future OAEI challenge. Further refinement of word vectors will be explored through the use of synonym and antonym relationships to learn semantic representation of words through the use of thesaurus such as Wordnet.

References

[1] D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm, "Schema and ontology matching with coma++," in *ACM SIGMOD Intl. Conference on Management of Data*, 2005, pp. 906–908.

[2] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy, "Learning to match ontologies on the semantic web," *The VLDB journal*, vol. 12, no. 4, pp. 303–319, 2003.

[3] A. Doan, A. Halevy, and Z. Ives, *Principles of Data Integration*. Elsevier, 2012.

[4] M. Lenzerini, "Data integration: A theoretical perspective," in *ACM SIGMOD Symposium on Principles of Database Systems*, 2002, pp. 233–246.

[5] L. D. Xin, "Big data integration (synthesis lectures on data management)," 2015.

[6] "Oaei," *http://oaei.ontologymatching.org/*, 2020.

[7] A.-C. Ngonga Ngomo, M. A. Sherif, K. Georgala, M. Hassan, K. Dreßler, K. Lyko, D. Obraczka, and T. Soru, "LIMES - A Framework for Link Discovery on the Semantic Web," *German Journal of Artificial Intelligence*, 2021.

[8] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, and F. M. Couto, "The agreementmakerlight ontology matching system," in *OTM Intl. Conferences On the Move to Meaningful Internet Systems*. Springer, 2013, pp. 527–541.

[9] P. Oram, "Wordnet: An electronic lexical database. christiane fellbaum (ed.)," *Applied Psycholinguistics*, vol. 22, no. 1, p. 131, 2001.

[10] Z. Wu and M. Palmer, "Verb semantics and lexical selection," *arXiv preprint cmp-lg/9406033*, 1994.

[11] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," *arXiv preprint cmp-lg/9511007*, 1995.

[12] D. Lin *et al.*, "An information-theoretic definition of similarity." in *Intl. Conference on Machine Learning (ICML)*, vol. 98, no. 1998, 1998, pp. 296–304.

[13] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," *arXiv preprint cmp-lg/9709008*, 1997.

[14] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "S-match: An algorithm and an implementation of semantic matching," in *European Semantic Web Symposium*. Springer, 2004, pp. 61–75.

[15] L. Serafini, P. Bouquet, B. Magnini, and S. Zanobini, "An algorithm for matching contextualized schemas via sat," 2003.

[16] C. Pesquita, C. Stroe, I. F. Cruz, and F. M. Couto, "Blooms on agreementmaker: Results for oaei 2010," *Ontology Matching*, p. 134, 2010.

[17] E. Jiménez-Ruiz and B. C. Grau, "Logmap: Logic-based and scalable ontology matching," in *Intl. Semantic Web Conference*. Springer, 2011, pp. 273–288.

[18] F. M. Suchanek, S. Abiteboul, and P. Senellart, "Paris: Probabilistic alignment of relations, instances, and schema," *arXiv preprint arXiv:1111.7164*, 2011.

[19] J. Li, J. Tang, Y. Li, and Q. Luo, "Rimom: A dynamic multistrategy ontology alignment framework," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 8, pp. 1218–1232, 2008.

[20] W. Hu, J. Chen, and Y. Qu, "A self-training approach for resolving object coreference on the semantic web," in *The World Wide Web Conference*, 2011, pp. 87–96.

[21] J. Gracia, J. Bernad, and E. Mena, "Ontology matching with cider: evaluation report for oaei 2011," *Ontology Matching*, p. 126, 2011.

[22] D. Ngo and Z. Bellahsene, "Yam++: a multi-strategy based approach for ontology matching task," in *Intl. Conference on Knowledge Engineering and Knowledge Management*. Springer, 2012, pp. 421–425.

[23] L. Bühmann, J. Lehmann, P. Westphal, and S. Bin, "Dl-learner structured machine learning on semantic web data," in *The Web Conference*, 2018, pp. 467–471.

[24] Y. Zhang, X. Wang, S. Lai, S. He, K. Liu, J. Zhao, and X. Lv, "Ontology matching with word embeddings," in *Chinese computational linguistics and natural language processing based on naturally annotated big data*. Springer, 2014, pp. 34–45.

[25] L. L. Wang, C. Bhagavatula, M. Neumann, K. Lo, C. Wilhelm, and W. Ammar, "Ontology alignment in the biomedical domain using entity definitions and context," *arXiv preprint arXiv:1806.07976*, 2018.

[26] G. Li, "Deepfca: Matching biomedical ontologies using formal concept analysis embedding techniques," in *Intl. Conference on Medical and Health Informatics*, 2020, pp. 259–265.

[27] V. Iyer, A. Agarwal, and H. Kumar, "Veealign: A supervised deep learning approach to ontology alignment," *Ontology Matching*, p. 216, 2020.

[28] J. Chen, E. Jiménez-Ruiz, I. Horrocks, D. Antonyrajah, A. Hadian, and J. Lee, "Augmenting ontology alignment by semantic embedding and distant supervision," in *European Semantic Web Conference*. Springer, 2021.

[29] Z. Wang, Q. Lv, X. Lan, and Y. Zhang, "Cross-lingual knowledge graph alignment via graph convolutional networks," in *Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 349–357.

[30] "Anatomy," *http://oaei.ontologymatching.org/2019/anatomy/index.html*, 2013.

[31] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[32] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[33] A. Lerer, L. Wu, J. Shen, T. Lacroix, L. Wehrstedt, A. Bose, and A. Peysakhovich, "Pytorch-biggraph: A large-scale graph embedding system," *arXiv preprint arXiv:1903.12287*, 2019.

[34] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Neural Information Processing Systems (NIPS)*, 2013, pp. 1–9.

[35] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of Machine Learning Research*, vol. 12, no. 7, 2011.

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[37] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[38] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint 1810.04805*, 2018.

[39] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[40] F. Crestani and C. J. van Rijsbergen, "Information retrieval by logical imaging," *Journal of Documentation*, vol. 51, no. 1, pp. 3–17, 1995.

[41] "Largebio," *http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/*, 2020.