VeeAlign: A Supervised Deep Learning Approach to Ontology Alignment*

Vivek Iyer[1], Arvind Agarwal[2], and Harshit Kumar[2]

[1] IIIT Hyderabad, India
vivek.iyer@research.iiit.ac.in
[2] IBM Research, India
{arvagarw,harshitk}@in.ibm.com

Abstract. [3]While deep learning approaches have shown promising results in Natural Language Processing and Computer Vision domains, they have not yet been able to achieve impressive results in Ontology Alignment, and have typically performed worse than rule-based approaches. Some of the major reasons for this are: a) poor modelling of context, b) overfitting of standard DL models, and c) dataset sparsity, caused by class imbalance of positive alignment pairs wrt negative pairs. To mitigate these limitations, we propose a dual-attention based approach that uses a multi-faceted context representation to compute contextualized representations of concepts, which is then used to discover semantically equivalent concepts.

Keywords: Ontology Alignment · Deep Learning.

1 Presentation of the System

The task of ontology alignment aims to determine correspondences between semantically related concepts - classes and properties- across two ontologies. OAEI [3] has been conducting ontology alignment challenges since 2004 where multiple datasets belonging to different domains are released along with a public evaluation platform to evaluate different matching systems. Matching systems that have been proposed can broadly be classified into two types: rule based systems [4, 7] and statistical based systems [5, 8, 9, 11]

Rule based system uses handcrafted rules with manually assigned weights, coupled with various string similarity algorithms to discover concept alignments, often utilizing domain-specific knowledge as well. This kind of approach, while easy to implement, has some obvious limitations: a) Using string similarity algorithms with minimal focus on context does not address either semantic or contextual relatedness. b) For every pair of ontologies, a new set of rules and weights may need to be defined, which is often a laborious and time consuming process, thus adversely affecting scalability.

* This work was done while Vivek Iyer was an intern at IBM Research, India

[3] Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Deep Learning models proposed so far [11] have used external information to enrich entities, such as synonyms from the ontology, definitions from Wikipedia and context from background knowledge sources. Along similar lines, DeepAlign[9] uses external information, such as synonyms and antonyms extracted from WordNet and PPDB, for refining word vectors, which are then used for alignment. Despite the uniqueness of the proposed approaches, Deep Learning models have typically been unable to thrive in the task of Ontology Alignment, and have performed worse than rule-based models. Some of the major reasons for this are: a) lack of focus on ontological structure, and thus poor modelling of context, b) overfitting of standard DL models, and c) dataset sparsity, caused by class imbalance of positive alignment pairs wrt negative pairs. This occurs because the number of positive alignments is typically several orders smaller than the number of negative alignments.

1.1 State, Purpose and General Statement

In an effort to mitigate the above-mentioned challenges, we propose VeeAlign[6], an ontology alignment system that aligns classes and properties based on not just semantic, but also structural similarity which is driven by its context. Our method thus incorporates a novel way of modelling context, where it is split into multiple facets based on the type of neighborhood. Based on their relative importance, some of these facets include only neighbouring one-hop nodes, while others also include the paths from the root to these nodes. To address this challenge, we use a novel dual attention mechanism that comprises of path level attention followed by node level attention. The former helps find the most important path among all the available paths, while the latter finds the node with the greatest influence on the alignment of the central concept.

1.2 Specific Techniques Used

VeeAlign[6] is a supervised Deep Learning based ontology alignment system, that computes a contextualized representation of concepts as a function of not just its label but also its multi-faceted neighbouring concepts surrounding it. In other words, the context is divided into multiple facets based on the relationship between the concept and its neighbours, and then a contextual vector is computed using a dual attention mechanism. This contextual vector is later concatenated with semantic distribution of the concept label, thus computing a contextualised concept representation which is later used to discover alignments. Figure 1 shows the architecture diagram of the proposed system. Thus, the key significance of the proposed approach is that VeeAlign exploits not just the semantic aspect, like previous approaches, but also the syntactical structure of ontologies and uses that alone to achieve state-of-the-art results, without any requirement for background knowledge.

Context Representation In VeeAlign, in the case of concepts, its neighboring concepts form its context. Each neighboring concept has a role and exerts a

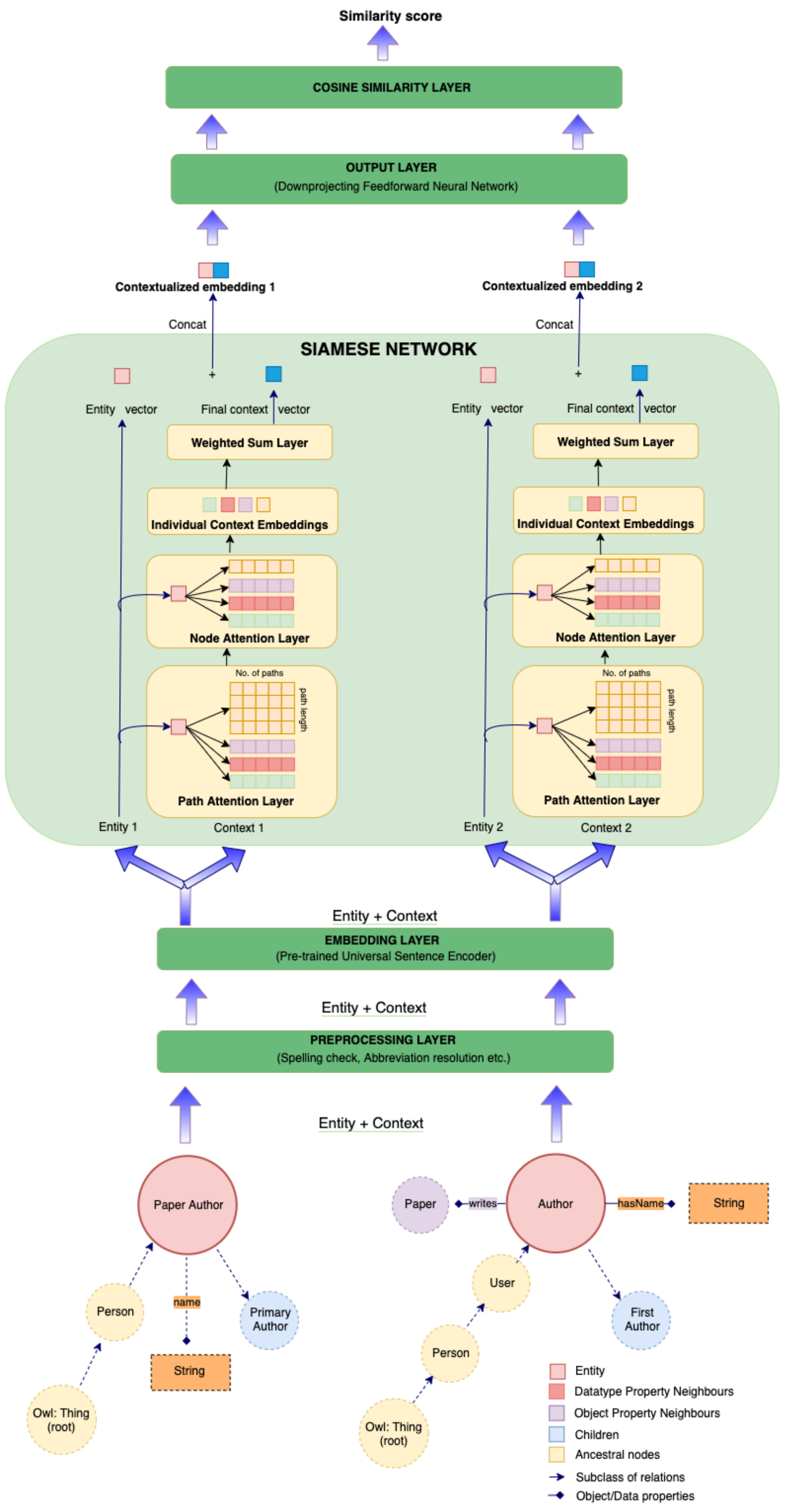

Fig. 1: VeeAlign Architecture

separate influence on the concept alignment, therefore we categorize neighboring concepts into four facets: ancestor nodes, child nodes, nodes connected through a datatype property and nodes connected through an object property. Here, we define parent and child nodes as those connected to the central node through SUBCLASS-OF relations. Sifting through several ontologies and their reference alignments, we observed that two concepts align not just based on their one-hop neighbours, but also on the basis of similarity of "ancestral nodes". Ancestral nodes of a concept are defined as the nodes that lie on the path all the way from the root to the parent node of the concept, and this path is referred to as a "lineage path". Thus, given a concept, apart from its one-hop neighbours we also enumerate all its lineage paths, consider them as part of context and use them while computing alignments. Since only ancestors will have such "lineage paths" and not one-hop neighbours (children, datatype property neighbours and object property neighbours), we consider these one-hop neighbours as paths of length 1 in order to maintain consistent terminology. The context of properties is modelled separately, by considering their domain and range as context.

Dual Attention Attention [1, 10] in deep learning can be broadly interpreted as a vector of weights denoting relative importance. For the task at hand, attention weighs neighboring concepts in proportion with their influence on the central concept's alignment. The attention process used to compute weights consists of a dual attention mechanism: first at the path level, referred to as Path-level attention, and the next at the node level, referred to as Node-level attention. The goal is to assign higher weights to the more influential paths using Path-level attention. And, within the most influential path, higher weights are assigned to nodes that are the most influential.

Thus, path level attention aims to find the most important paths for each contextual facet. This involves computing the attention weight of each node in each path with respect to the main concept, and then adding and normalizing these weights. When done for each lineage path, it yields the relative importance of that path. Given the relative importance of each path, a max-pool layer is then applied over each of these path weights to yield the most important path. After path-level attention, the next step is node-level attention. This is achieved by computing the attention weights of each node in the most important path. These weights are then used to take a weighted linear combination of the node embeddings available in the path embedding.

Model Training As shown in Figure 1, the training process involves computing the representations of all four types of context i.e., ancestral context, child context, data property neighbour context and object property neighbour context. The context representation computation involves applying the path-level attention followed by the node-level attention. The child, data properties and object properties context have paths of only length one, therefore there is only path level attention, not the node level attention. These four context representations are combined through a weighted linear combinations to get the final context representation. This context vector is then concatenated with the semantic representation of the central concept, and the combined representation is input to a linear layer for dimensionality reduction in a lower dimension space. Since a candidate alignment pair consists of elements (concepts or properties) from both source and target ontologies, we perform the above computations for both source and target elements to get the contextualized representations by passing both through a Siamese Network [2], and then compute the confidence score of the alignment by taking a cosine similarity between the two contextualized representations. For the alignment of properties, we consider its context as its domain and range, and obtain the confidence score as a weighted sum of the similarities between the respective distributional embeddings of domains, ranges and property labels respectively. Finally, an element pair is considered as a positive alignment when the similarity score is more than an experimentally determined threshold, and discarded otherwise. We use mean squared loss computed between the predicted and ground truth labels for training our model.

1.3 Datasets and Experimental Setting

Our system requires training data in the form of positive and negative alignment pairs. Although many tracks in OAEI have reference alignments that contain positive example pairs, as per the rules, one could not use them for training. So, we resorted to using pseudo-training data, i.e the output of the highest-performing system, AML, as an approximation of the reference alignments. However, we found that AML was only able to identify concepts whose names had some sort of string similarity, but not concepts that had different names but similar contexts. Our approach of modelling structural context and then attending on it thrives on identifying not just the former using semantic similarity, but also the latter using structural similarity. Since this was missing in AML's output, there were a large number of False Negatives (FNs). In addition, there were a few False Positives (FPs) with similar names but different contexts. Thus instead of using AML's output directly, we decided to manually correct AML's output as a preliminary stage. In order to discover FPs, we simply validated AML's output and discovered 12 FPs. To discover FNs, we took a cartesian product of all concept pairs, sorted them based on context similarity and annotated the top 1000 pairs, discovering a total of 35 FNs. Our annotations process included three annotators where the final decision was taken using majority voting algorithm.

As part of our submission, we targeted two tracks i.e. Conference and Multifarm. We performed the above exercise for all ontology pairs in the Conference track. For Multifarm, due to lack of time, we could not train a separate model or integrate a translator, but instead used the pre-trained Conference model, since both tracks share similar ontological structure and are comprised of the same ontologies, albeit in different languages. In place of a translator, we merely

Table 1: Conference track results of OAEI 2020

	ra1-M1	ra1-M2	ra1-M3	ra2-M1	ra2-M2	ra2-M3	rar2-M1	rar2-M2	rar2-M3
VeeAlign	**0.78**	0.34	0.73	**0.74**	0.34	0.69	**0.74**	0.34	**0.7**
AML	0.76	**0.58**	**0.74**	0.71	**0.58**	**0.7**	0.71	**0.56**	0.69
LogMap	0.73	0.39	0.69	0.67	0.39	0.63	0.69	0.4	0.66
Wiktionary	0.69	0.26	0.61	0.64	0.26	0.57	0.65	0.27	0.58
ATBox	0.69	0.23	0.6	0.64	0.26	0.56	0.65	0.26	0.57
ALOD2Vec	0.68	0.25	0.59	0.62	0.25	0.55	0.64	0.26	0.56
LogMapLt	0.66	0.23	0.59	0.6	0.23	0.54	0.62	0.23	0.56
ALIN	0.67	-	0.6	0.61	-	0.55	0.63	-	0.56
edna	0.67	0.14	0.59	0.61	0.14	0.54	0.63	0.14	0.56
StringEquiv	0.64	0.03	0.56	0.58	0.03	0.52	0.61	0.03	0.53
Lily	0.62	-	0.55	0.57	-	0.5	0.57	-	0.51
DESKMatcher	0.25	0.02	0.18	0.23	0.02	0.16	0.23	0.02	0.16

substituted the initial Universal Sentence Encoder[4] model with its multilingual variant[5].

Our Deep Learning model requires several parameters and hyperparameters, which we optimized through a grid-search algorithm. The model was converged using MSE loss and Adam optimizer with a learning rate of 0.001, after training for 50 epochs with a batch size of 32. The maximum number of paths considered for each node is set to 21, and the maximum length of the path is taken as 8. All randomizations including the ones in PyTorch and Numpy are done by setting 0 as the manual seed.

1.4 Link to the System and Parameters File

The entire codebase of our system is available on GitHub[6] and so is the configuration parameters file[7].

2 Results and Discussions

2.1 Conference

Table 1 shows the results of the conference track of OAEI 2020. There are in total 9 matching tasks. There are three different reference alignments: ra1 (original), ra2 (consistency violation-free) and rar2 (consistency & conservativity violation-free), each of which contain class matching, property matching and hybrid (both class and property) matching tasks respectively. VeeAlign achieves state-of-the-art results in all the entity matching tasks, 3rd position in all the property matching tasks and top 2 positions in all the hybrid matching tasks, sometimes losing out to AML by a narrow margin of 0.01 points in F1-score. In rar2, which

[4] https://tfhub.dev/google/universal-sentence-encoder-large/5
[5] https://tfhub.dev/google/universal-sentence-encoder-multilingual/3
[6] https://github.com/Remorax/VeeAlign
[7] https://github.com/Remorax/VeeAlign/blob/master/src/config.ini

Table 2: Multifarm track results of OAEI 2020

System	Different ontologies	Same ontologies
AML	**0.47**	0.28
LogMap	0.37	**0.41**
VeeAlign	0.15	0.14
Wiktionary	0.32	0.12
LogMapLt	0.04	0.01

has been considered to be the main set of reference alignments this year (since it is both consistency and conservativity violation-free), VeeAlign tops the table.

2.2 Multifarm

Table 2 shows the multifarm track results. As expected, VeeAlign is unable to compete with state-of-the-art systems in this track, but still manages to produce decent results when one factors in the lack of a separately trained model and more importantly, a translator.

3 General comments

3.1 Comments on results

VeeAlign has certainly produced some very encouraging results. In the conference track, it achieves state-of-the-art results in entity matching, which indicates that our two-step attention model that adopts a multifaceted approach to context representation, is able to outperform systems that use handcrafted rules and manually assigned parameter values. VeeAlign's performance dips when it comes to property matching, indicating that our approach of modelling property similarity as a weighted sum of property, domain and range similarity is not good enough and needs more work. Nevertheless, our current approach ensured we achieve state-of-the-art results in hybrid-matching tasks, a sizeable feat in just the debut run.

In the multifarm track, unsurprisingly VeeAlign found itself being unable to compete with other SOTA systems, that, unlike VeeAlign, used an integrated translator. However despite this and the lack of a separately trained model, it was still able to use ontological structure to produce moderately decent results with low recall but high precision.

3.2 Comments on OAEI measures

While current OAEI evaluation measures are definitely thorough and multifaceted, a possible improvement would be to incorporate K-fold sliding window evaluation. Here, (K-1) folds of the reference alignments are provided as input (if the system needs it for training) and the last fold is used for testing. This is done for every possible fold, and an average of the performance on each fold is taken to yield the final performance. This data split into K-folds can occur either at the ontology-pair level (possible in tracks where there are a large number of ontology pairs in the reference alignments, such as conference or multifarm tracks) or at the "concept-pair" level (possible in tracks where there are fewer number of ontology pairs, but the ontologies are larger in size). In case of the former, (K-1) folds of ontologies are provided for training and 1 fold for testing. In the conference track, for instance, which has 21 ontology pairs , if we take K=7, 6 folds (i.e 18 ontology pairs) are provided for training and the last fold (i.e 3 ontology pairs) are tested on. This is repeated for every 3 pairs of ontologies, and an average is taken.

In case of the latter, a "concept-pair" split could be achieved by taking a cartesian product of the concepts in both ontologies, splitting it into K folds, taking K-1 for training and in each test fold, the concept pairs which are part of the alignment output by a system can be marked as True, and the rest as False. The same can be done for the ground truth alignments, and the corresponding True Positives (TPs), False Positives (FPs) and True Negatives (TNs) can be calculated to yield precision, recall and F1-scores. When evaluating by "concept-pair" split, the input for non-supervised systems remains the same, and is still the entirety of the ontology. For supervised systems, it would be the (K-1) folds of concept pairs, with pairs present in reference alignments being marked as True and the rest as False. Such a file could be dynamically created and input to the system. However, while evaluating the alignments generated by each system, the procedure remains the same, i.e calculating TPs, FPs and TNs on the last fold.

Both of these proposed methods of evaluation are in compliance with OAEI rules, and the training and testing data are clearly separated (removing any chance of over-fitting) and an average is taken to remove any chance of bias. Moreover, K-fold sliding window evaluation is widely accepted in the AI community as a fair mode of evaluation. Going ahead, these measures would give supervised systems a fair chance to compete against systems that use handcrafted rules, would encourage more DL-based systems in the future and thus allow Deep Learning to thrive in Ontology Alignment as well.

4 Conclusion & Future Work

As part of OAEI 2020, we introduced VeeAlign, an Ontology Alignment that uses a supervised Deep Learning approach to discover alignments. In particular, it uses a two-step model of attention combined with multi-facted context representation to produce contextualized representations of concepts, which aids alignment based on semantic and structural properties of an ontology. VeeAlign achieves state-of-the-art results in Conference track beating AML, LogMap and other mature systems in just its debut run, which is certainly encouraging and indicative of the validity and effectiveness of our approach. In multifarm track, VeeAlign produces decent results using solely ontological structure. The results have high precision but low recall due to lack of incorporation of a translator, which we plan to fix in future editions of OAEI. In addition, we plan on improving property matching to further improve our performance on the Conference track. Lastly and mostx importantly, we plan on targeting the biomedical tracks (such as Anatomy, LargeBio and Biodiversity) and adapting VeeAlign to use biomedical background knowledge, if available.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. ICLR (2015)
2. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a" siamese" time delay neural network. In: Advances in neural information processing systems. pp. 737–744 (1994)
3. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology alignment evaluation initiative: six years of experience. In: Journal on data semantics XV, pp. 158–192. Springer (2011)
4. Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I.F., Couto, F.M.: The agreementmakerlight ontology matching system. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 527–541. Springer (2013)
5. Huang, J., Dang, J., Vidal, J.M., Huhns, M.N.: Ontology matching using an artificial neural network to learn weights. In: IJCAI workshop on semantic Web for collaborative knowledge acquisition. vol. 106 (2007)
6. Iyer, V., Agarwal, A., Kumar, H.: Multifaceted context representation using dual attention for ontology alignment. arXiv preprint arXiv:2010.11721 (2020)
7. Jiang, S., Lowd, D., Kafle, S., Dou, D.: Ontology matching with knowledge rules. In: Transactions on Large-Scale Data-and Knowledge-Centered Systems XXVIII, pp. 75–95. Springer (2016)
8. Jiménez-Ruiz, E., Agibetov, A., Chen, J., Samwald, M., Cross, V.: Dividing the ontology alignment task with semantic embeddings and logic-based modules. arXiv preprint arXiv:2003.05370 (2020)
9. Kolyvakis, P., Kalousis, A., Kiritsis, D.: Deepalignment: Unsupervised ontology matching with refined word vectors. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 787–798 (2018)
10. Paulus, R., Xiong, C., Socher, R.: A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304 (2017)
11. Wang, L., Bhagavatula, C., Neumann, M., Lo, K., Wilhelm, C., Ammar, W.: Ontology alignment in the biomedical domain using entity definitions and context. In: Proceedings of the BioNLP 2018 workshop. pp. 47–55 (2018)