GraphMatcher: A Graph Representation Learning Approach for Ontology Matching

Sefika Efeoglu

Free University of Berlin, Department of Computer Science, Takustraße 9, 14195 Berlin,Germany

Abstract

Ontology matching might be defined as finding a relationship or correspondence between two or more entities in two or more ontologies. To solve the interoperability problem of the domain ontologies, semantically similar entities in these ontologies must be found and aligned before merging them. GraphMatcher, developed in this study, is an ontology matching system using a graph attention approach to compute a higher-level representation of a class together with its surrounding terms. The GraphMatcher has obtained remarkable results in OAEI 2022 conference track. Its codes are available https://github.com/sefeoglu/gat_ontology_matching.

Keywords

graph attention, graph representation, ontology matching

1. Presentation of the system

GraphMatcher is a new ontology matching system based on graph representation learning using a graph attention [1] together with a new neighborhood aggregation approach. The graph representation learning approach has leveraged the graph attention and introduces a new neighborhood aggregation algorithm that increases contextual information of the center class and property.

1.1. State, propose, general statement

Ontology matching has been defined as finding a relationship or correspondence between two or more entities in two or more independent ontologies. The alignments of two ontologies are classified as two different cases: (i) simple alignment and (ii) complex alignment [2]. The simple alignment is defined as the mapping of the class names according to word-based similarity, while complex alignment considers the meaning of two classes to decide whether they are similar [2]. To understand the meaning of a class (sequence), the contextual information of a class or property is needed. In this case, we have to decide which neighbor contributes to their contextual information.

Many logic- and algorithm-based ontology matching tools, such as LogMAP [3] and AML [4] solve this interoperability problem of domain ontologies by using algorithms and logic-based

The 17th International Workshop on Ontology Matching, The 21th International Semantic Web Conference ISWC-2022, October 23rd, 2022, Hangzhou, China

sefika.efeoglu@fu-berlin.de (S. Efeoglu)

approaches. In addition to these approaches, DeepAlignment [5], VeeAlign [6], and Convolutional Networks of *Bento et al. (2020)* [7] apply ML for matching. Nevertheless, according to OAEI's [1] conference track results, it is reported that they cannot achieve a better performance than traditional tools like AML and LogMAP. The weakness of these ML approaches might be due to the lack of contextual information about a property and class. Another limitation is in how to represent the ontology's data as a convolutional graph like an image, for example where each pixel in the image data has the same number of its neighboring pixels, whereas each class in ontology has a different number of its neighboring terms like an arbitrary graph [1]. The most appropriate way of representing the data in the ontology is the arbitrary graph.

Since the ontology represents the data with an arbitrary graph, we aim to develop a graph representation learning model based on a graph attention mechanism [1] using Siamese networks [7, 8, 6] to find the semantically similar concepts within the ontologies. The graph attention mechanism computes the higher-level representation of a concept and its surrounding concepts (features). The model then finds similarity scores between the concept pairs among the aligned ontology pairs and decides the concept alignments.

1.2. Specific Techniques Used

GraphMatcher utilizes a graph representation learning approach using a graph attention [1]. Its network consists of five layers. The main contribution is adaptation of graph attention approach to the Siamese network in the third layer.

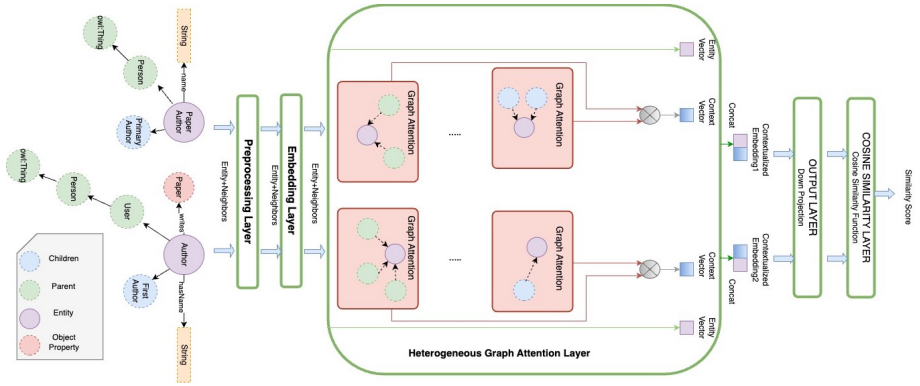

Figure 1: The proposed network is an application of heterogeneous graph attention (GAT) on Siamese Networks to find the similarity score of classes.

1.2.1. Preprocessing

Data preprocessing is one of the most significant parts of developing a machine learning model and is required to explain the variability of features in a sample. In this study, we have handled data preprocessing of an ontology in six steps (i) an ontology parsing, (ii) tokenization, (iii)

[1]Ontology Alignment Evaluation Initiative (OAEI):
http://oaei.ontologymatching.org/

finding the abbreviations, (iv) cleaning from stop words, (v) neighborhood aggregation for creating the context, and (vi) finding the embedding of the terms.

1.2.2. Embedding Layer

We use the pre-trained Universal Sentence Encoder (USE) to obtain the word embedding vector of a class and property. According to benchmark results [9], the USE outperforms the BERT encoder on semantic sentence encoding. Therefore, we have preferred the USE for word embeddings.

1.2.3. Heterogeneous Graph Attention Layer

The graph attention introduced in Graph Attention Networks is used to cluster and classify the citation graphs [1]. It provides inductive and transactive learning approaches. Its inductive learning approach computes contextual information by applying an attention centered to the center class in the neighborhood graph. The center class has five homogeneous graphs consisting of its neighboring terms, and these five graphs refer to one heterogeneous graph.

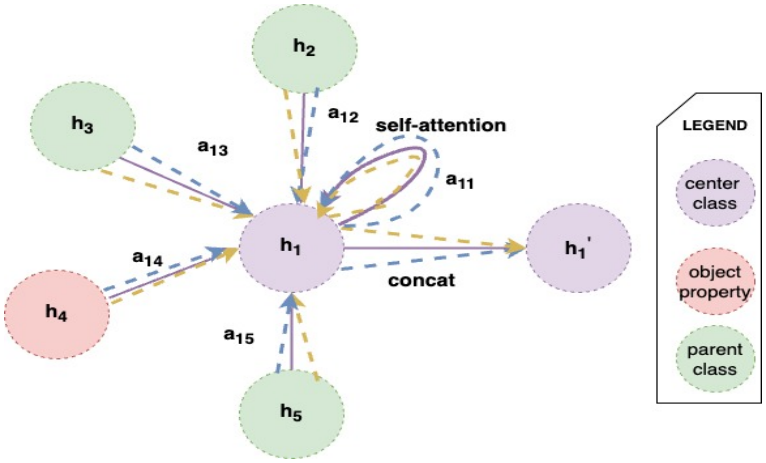

Figure 2: This figure shows how graph attention is applied to a homogenous graph. The relationship between the center class and its neighboring classes is the same in the homogenous graphs like 'subClassOf'. In this graph, the 'ObjectProperty' node represents the restrictions belonging to the definition of a child or parent class. Bag-Of-Word is used to represent a center class and its neighboring classes. $h_1^{'}$ shows the contextual information of h_1 in terms of this 'subClassOf' relation after applying graph attention. In addition, a_{ij} denotes α_{ij} in Equation 5.

In addition to this, the relation between the center class and other terms in each homogeneous graph is the same; for example, the relation between the center node and other terms in Figure 2 is 'subClassOf'. However, some homogeneous graphs consisting of 'equivalentClass' or 'subClassOf' relationships might contain some terms regarding restrictions defined with 'ObjectProperty' as stated in the neighborhood aggregation of the data preprocessing section. The figure explains how a center class' contextual information is computed by graph attention

in one of its homogenous neighborhood graphs. The main difference between the original graph attention approach and our graph attention is that we apply this attention mechanism to a heterogeneous neighborhood graph containing five homogeneous graphs. The attention in this layer covers the following various attention mechanisms [1].

A set of features (center class and its neighbouring terms), which is inputs of the graph attention layer, is denoted in Eq. 1 where $\vec{h_i} \epsilon \mathbb{R}^F$.

$$h = \left\{ \vec{h_1}, \vec{h_2}, \vec{h_3} ..... \vec{h_N}, \right\} \tag{1}$$

The layer convert the input features to the new higher level representation of the feature list like defined in Eq. 2 where $\vec{h_i} \epsilon \mathbb{R}^{F'}$

$$h^{'} = \left\{ \vec{h_1^{'}}, \vec{h_2^{'}}, \vec{h_3^{'}} ..... \vec{h_N^{'}}, \right\} \tag{2}$$

Using features in Eq. 1, we have obtained the higher-level representation of a class' neighbors, namely its contextual information in this layer. The new features are computed by Eq. 3, and σ indicates the linear activation function like sigmoid or softmax in the Eq. 3. To indicate the higher-level representation of the set of features, $W \epsilon \mathbb{R}^{F' X F}$ is used as a learnable parameter, and shared linear transformation is applied to each feature. "K" is the number of independent heads, and "K" equals five in our project.

$$h' = ||_{k=1}^{K} \sigma \left(\sum \alpha_{ij}^{k} W^k \vec{h_j} \right) \tag{3}$$

$\alpha \epsilon \mathbb{R}^{F'} X \mathbb{R}^{F'}$ denotes a shared attention mechanism and a layer using the self-attention. The following equation computes attention coefficients (e_{ij}):

$$e_{ij} = \alpha(W\vec{h_i}, W\vec{h_j}) \tag{4}$$

To compute the α_{ij} for h' in the Eq. 2, this coefficient attention mechanism is applied in Eq. 5 where $\vec{a} \epsilon \mathbb{R}^{2F}$ and α_{ij} is attention mechanism parameterized by $\vec{a}$ weight vector.

$$\alpha_{ij} = softmax(e_{ij}) = \frac{\exp(LeakyReLU(\vec{a}^T [W\vec{h_i} || W\vec{h_j}]))}{\sum_{k \epsilon N_i} \exp(LeakyReLU(\vec{a}^T [W\vec{h_i} || W\vec{h_k}]))} \tag{5}$$

Using the formula in the Eq. 3 in this layer, we have obtained the higher-level representation of a class' neighbors, namely its contextual information in this layer.

1.2.4. Output Layer

Output Layer provides the down sampling (dimensional reduction) of the contextual information, which is the concatenation of class embedding and higher representations of the class's neighbors.

1.2.5. Cosine Similarity Layer

Cosine Similarity Layer measures the cosine similarity of the output of the previous layer.

1.3. Adaptations made for the evaluation

The GraphMatcher's framework has been developed in Pyhton with pytorch and ontospy, and is packed by SEALS using MELT. We submitted the model which gives the highest micro-F1 when evaluated on the local reference alignments with MELT.

1.4. Parameter Settings

After 5-fold cross-validation of the network on the conference track, 0.001 of learning rate, 5 epochs, 0.001 of weight decay and 32 of batch size have given the optimum results on the model. The threshold is computed from False Positive alignment in the validation data as how the VeeAlign [6] system proposes [2].

2. Results

Conference data set consists of sixteen ontologies, but only seven ontologies have reference alignment cases (twenty-one reference alignment). These reference alignments has been utilized as ground truths to use the possible true positive alignment cases. Besides, the negative alignment cases has been computed by oversampling from the all possible class and property alignments. Therefore, once we apply the supervised machine learning approach, we can use only these seven ontologies on this data set.

Table 1
The table shows GraphMatcher results in the Conference track.

Evaluation	Precision	F.5-measure	F1-measure	F2-measure	Recall
ra1-M1	0.82	0.77	0.71	0.65	0.62
ra1-M2	0.65	0.51	0.39	0.32	0.28
ra1-M3	0.8	0.74	0.67	0.6	0.57
ra2-M1	0.78	0.73	0.66	0.6	0.57
ra2-M2	0.65	0.5	0.39	0.32	0.28
ra2-M3	0.76	0.7	0.62	0.56	0.53
rar2-M1	0.77	0.73	0.67	0.62	0.59
rar2-M2	0.65	0.52	0.4	0.33	0.29
rar2-M3	0.75	0.7	0.63	0.58	0.55

The table 1 shows the GraphMatcher results in the conference track. The performance of the GraphMatcher is better in M1 variants than in M2 in terms of F1-measure. Therefore, its F1-measure has been decreased in M3 variants. As a result, the model has weakness in the M2 variants, namely property alignment.

[2]The project uses directly VeeAlign's approach to compute the threshold with the permission of the fist author.

3. General Comments

3.1. Comments on the results

The GraphMatcher is a new ontology matching system participating in OAEI 2022 and is evaluated in the conference track. The GraphMatcher demonstrates remarkable performance in the M1 and M3 evaluation variants in terms of F1-measure, even though it does not obtain high results in M2 evaluation variants. However, all other matchers do not show remarkable results in that evaluation case.

On the other hand, it is also evaluated in the uncertain reference alignments in OAEI 2022's conference track. It has the highest F1-measure (72% in both of them) in the discrete and continuous metrics [3] among all other matchers evaluated in this track. This means that the GraphMatcher's confidence is higher than the other matchers evaluated in the OAEI 2022 conference track.

Table 2
The table shows the matchers' performances on the rar2-M3 reference alignments.

Matcher	Precision	F.5-measure	F1-measure	F2-measure	Recall
LogMap	0.76	0.71	0.64	0.59	0.56
GraphMatcher	0.75	0.7	0.63	0.58	0.55
SEBMatcher	0.79	0.7	0.6	0.52	0.48
ATMatcher	0.69	0.64	0.59	0.54	0.51
ALIN	0.82	0.7	0.57	0.48	0.44
LogMapLt	0.68	0.62	0.56	0.5	0.47
LSMatch	0.83	0.69	0.55	0.46	0.41
AMD	0.82	0.68	0.55	0.46	0.41
KGMatcher+	0.83	0.67	0.52	0.43	0.38
ALIOn	0.66	0.44	0.3	0.22	0.19
Matcha	0.37	0.2	0.12	0.08	0.07

3.2. Improvements

The GraphMatcher should be improved to match the properties since it does not perform well in the M2 evaluation variant. Its current version does not apply the graph attention approach to align the properties because of the lack of properties' neighbors, especially datatype properties. These object type and data type properties might not have enough neighborhood terms to construct contextual information in the ontology. In this case, the property's context might be improved with the external information in the next version of this model, and the graph attention can also be applied to align the properties.

[3]The evaluation based on the uncertain reference alignments: http://oaei.ontologymatching.org/2022/results/conference/index.html

4. Conclusion

In this study, we have introduced a new ontology-matching system called GraphMatcher. GraphMatcher adapted the graph attention to the homogenous subgraphs of the center class' neighbors to obtain contextual information about the center class. The graph attention has computed the higher level representation of each class and its surrounding classes and properties. The results demonstrate promising performances in M1 and M3 evaluation variants. The future work of this study will be to increase its performance in M2 evaluation variants.

References

[1] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks (2018). arXiv:1710.10903.

[2] É. Thiéblin, O. Haemmerlé, N. Hernandez, C. Trojahn, Survey on complex ontology matching, Semantic Web 11 (2020) 689–727.

[3] E. Jiménez-Ruiz, B. Cuenca Grau, Logmap: Logic-based and scalable ontology matching, in: L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, E. Blomqvist (Eds.), The Semantic Web – ISWC 2011, SBH, Berlin, Heidelberg, 2011, pp. 273–288.

[4] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, F. M. Couto, The agreementmakerlight ontology matching system, in: R. Meersman, H. Panetto, T. Dillon, J. Eder, Z. Bellahsene, N. Ritter, P. De Leenheer, D. Dou (Eds.), On the Move to Meaningful Internet Systems: OTM 2013 Conferences, SBH, 2013, pp. 527–541.

[5] P. Kolyvakis, A. Kalousis, D. Kiritsis, 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), ACL, 2018, pp. 787–798.

[6] V. Iyer1, A. Agarwal, H. Kumar, Veealign: A supervised deep learning approach to ontology alignment (2020).

[7] A. Bento, A. Zouaq, M. Gagnon, Ontology matching using convolutional neural networks, in: Proceedings of the 12th Language Resources and Evaluation Conference, ELRA, Marseille, France, 2020, pp. 5648–5653. URL: https://www.aclweb.org/anthology/2020.lrec-1.693.

[8] J. Chen, J. Gu, Adol: a novel framework for automatic domain ontology learning, Super Computing (2021).

[9] H. Hassan, G. Sansonetti, F. Gasparetti, A. Micarelli, J. Beel, Bert, elmo, use and infersent sentence encoders: The panacea for research-paper recommendation?, in: RecSys, WS, 2019.