AMD Results for OAEI 2022

Zhu Wang

[1]*ADVIS Lab, Dept of Computer Science*
University of Illinois at Chicago, Chicago IL 60607, USA

Abstract

AgreementMakerDeep (AMD) is a new flexible and extensible ontology matching system. It exploits the contextual and structural information of ontologies by infusing knowledge to pre-trained masked language model, and then filter the output mappings using knowledge graph embedding techniques. AMD learns from classes and their relations between classes by constructing vector representations into the low dimensional embedding space with knowledge graph embedding methods. The results demonstrate that AMD achieves a competitive performance in many OAEI tracks, but AMD has limitations for property and instance matching.

Keywords

Ontology matching, Knowledge graph embedding, pre-train language model

1. Presentation of the system

AgreementMakerDeep (AMD) is a new deep learning ontology matching system inspired by AgreementMaker [1, 2], AgreementMakerLight (AML) [3] and BootEA [4]. It is designed with the main goal of higher efficiency for ontology matching problems by applying pre-train language models and knowledge graph embedding methods. This year is the second time that AMD participates in OAEI.

1.1. State,purpose, general statement

Ontology matching aims to establish semantic correspondences or relationships between concepts or properties of different ontologies [5]. There is a wide range of algorithms developed for ontology matching, such as those that use lexical similarity with linguistic techniques [6], partition large ontology sets based on structural proximity [7], or detect graph similarity [8, 9]. However, such strategies may be time consuming [10], may use sparse and a high-dimensional training space [11], and may vary with the domains [12].

AMD mainly utilizes BERT-like pre-train language model for textual matching, but adopts the representative learning models [13, 14] to capture the relations as structural information with a translation vector between two classes.

OAEI'22: The 17th International Workshop on Ontology Matching, Oct 23–24, 2022, Hangzhou, China
zwang260@uic.edu (Z. Wang)
https://ellenzhuwang.github.io (Z. Wang)
0000-0001-6374-8735 (Z. Wang)

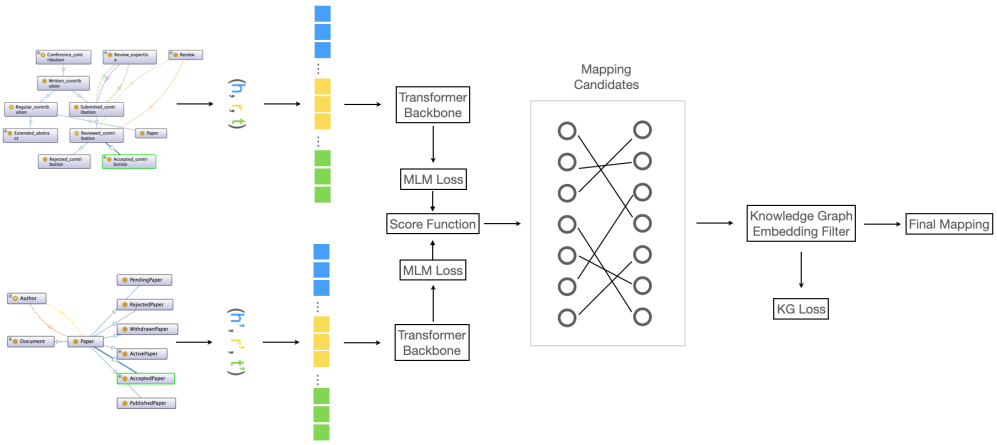

Figure 1: The overall framework of AMD. Here, inputs are sub-graphs of source and target ontologies. It learns language and knowledge embeddings by using a novel relation attention mechanism. Ontology matching tasks are solved using the outputs of masked language model, and final mappings are selected from candidate sets by knowledge graph embedding..

2. Specific Techniques Used

The architecture of AMD is shown in Fig. 1, including ontology parsing, textual matching with large pre-train language models, knowledge graph embedding, model learning and candidate selection.

Ontology parsing. owlready2 [15] is used to extract meta information of classes from the source and target ontology, such as super/sub-classes, labels, annotations, partof and disjointwith. BeautifulSoup [16] is used to extract synonyms.

Textual matching. We apply several text per-processing techniques like stop-words removal and tokenization on class labels and annotations. AMD uses sentence-BERT [17] to compute cosine similarity between two concept labels and annotations in unsupervised tasks. We consider to use textual matching results as our mapping candidations.

Knowledge graph embedding. We characterize the structure information of ontologies by relations translated from one class to another class using a modified TransR [18] model into relational embedding spaces.

2.0.1. Problem Formulation.

Given two ontologies O and O', we construct knowledge graph X and Y, and define the correspondence between two concepts as following triplets $T_{c,c'} = < c, r, c' >$, where r is the relation between c and c'. The problem is to find mapping set M = $\{(c_x, c_y) \epsilon X \times Y | c_x \equiv c_y\}$. In this study, we focus on one-to-one alignment and the relation between concepts is equality.

Let $\vec{v}(c_x)$= $\{v_1, v_2, ...v_m\}$ and $\vec{v}(c_y)$= $\{v'_1, v'_2, ...v'_n\}$ be two d-dimensional vectors sets of size m and n, we compute their distance with simple cosine similarity by d($\vec{v}(c_x), \vec{v}(c_y)$) = 1-sim($\vec{v}(c_x), \vec{v}(c_y)$) as follows:

$$sim(\vec{v}(c_x), \vec{v}(c_y)) = \sum_{i=1} \arg\max_j cos(\vec{v}(c_x), \vec{v}(c_y)) \quad (1)$$

We define the probability of the aligned labels between concepts c_x and c_y by $p(c_y|c_x)$ as follows:

$$p(c_y|c_x) = \sigma sim(\vec{v}(c_x), \vec{v}(c_y)) \tag{2}$$

where σ is the sigmoid function.

2.1. Masked Language Modeling

In bio-ML track, we train on ontologies corpus for semi-supervised tasks. We use standard transformer architecture[19] following Roberta[20]. The text encoder takes a sequence of tokens from triples $\{h, r, t\}$ as inputs, and computes a numbers of L layers to obtain contextualized representations $H_i \in \mathbb{R}^{N \times d}$, where N is the number of tokens in our vocabulary and d is the dimension.

Concept prediction Concepts are the dominant elements in ontology matching problem, therefore, predicting the concepts forces the model to learn the of semantic information. At the same time, we infuse structural knowledge by triples to enable the ability of the model to learn the contextualized representations for each ontology. Here, the concept prediction is to predict head or tail concept, and the difference is position embedding of the masked tokens.

For all the concepts $c \in C$, we randomly select 30% of them to predict. And for each selected concept c_i, the token w_i^h or w_i^t is replaced with the special token *[MASK]* in probability of 80%, another random token in 10% and the rest remain itself. The loss of concept prediction is defined as:

$$L_{c2r} = -\sum_{n+k}^{n+m+k} \log P(x_{n+k}|x_{<n+k}) \tag{3}$$

Relation prediction Relations express the way of connection head and tail concepts, and also provide enrich hierarchy as structural information. We considers to concatenate all tokens in r to predict, because relation labels usually have few words or tokens and are meaningless by separated tokens. For the training tasks, relation prediction is similar process as concepts prediction by masking randomly. Thus, the loss of relation prediction is written as:

$$L_{r2c} = -\sum_{n+m}^{n+m+k} \log P(x_{n+m}|x_{<n+m}) \tag{4}$$

Therefore, the masked language modeling loss function can now be written as,

$$L_{MLM} = L_{c2r} + \theta L_{r2c} \tag{5}$$

where we take a linear combination of both the loss terms.

2.2. Knowledge graph embedding

In AMD, we apply a modified TransR method which translates concepts and relations into concept space and relation-specify concept spaces, since there are multiple relations in the ontologies e.g subclassof and disjointwith. In the original TransR, the projected vectors are

defined as $c_r = cM_r, c'_r = c'M_r$, and the score function as $f_r(c, c') = \|c_r + r - c'_r\|_2^2$ [18]. Inspired by Sun et al. [4], the absolute scores of positive triples are lower than the negative ones, so we modify the loss function by using two γ hyper-parameters as follows:

$$\mathscr{L}_{KE} = \sum_{(T)\in S} \sum_{(T_{neg})\in S_{neg}} max(0, (f_r(T) - \gamma_1) - \mu(f_r(T_{neg}) + \gamma_2)) \tag{6}$$

where T denotes h, r, t and T_{neg} represent negative triples, $\gamma_1, \gamma_2, \mu > 0$ and $\gamma_2 > \gamma_1$. Negative triples are generated from negative sampling method by following AMD[14] and Multi-OM [6].

During the process that computes vectors, we need to generate negative triples. Following the work of Sun et al. [4] and Li et al. [6], we refine the uniform negative sampling by choosing from the k-nearest neighbors in the embedding space, and setting constraints of select candidates excluding from the subclassOf or disjointWith related concepts. In this way, we can avoid vector sparsity and obtain better quality of vector representations for the concepts.

Candidate selection We select candidates based on a threshold of the classes knowledge graph embedding vectors similarity, and then compare the similarity with baseline if the pairs are in baseline result sets.

2.3. Parameter settings

Ontology pre-processing. Ontologies always are in the format of owl or rdf, but the inputs of masked language model and knowledge graph embedding models require the format of word or token embeddings. Firstly, we extract meta information from ontologies using `owlready2` [1], such as ID, labels, resource, descriptions of class(or called concepts). The nature language information of relations are extracted from restriction, property, subClass or superClass. Since the ontologies in the tasks were developed by different organizations, we process the ontology parsing from different tags, e.g *rdf:ID="isPartOf"* and *rdf:resource ="UNDEFINED_part_of"*.

MLM pre-training settings. In practice, we use RoBERTa implementation by Huggingface[2] as the base pre-trained model in our all experiments. LaKERMap is initialized with the `roberta.base` parameters, and the base model size is 12 layers and 768-dimensional hiddden states($L = 12, d = 768$). For the MLM training task, we use the words or tokens in knowledge triples as our corpora for fine tuning. We select the first 5 mapping pairs from lexical matching method in few-shot learning. Hyper-parameters are the same in [20].

KGE training settings. We use the outputs of MLM as word or token embedding in the initialization for knowledge graph embbedding training process, as the dimension of d is set to 768. The remainder of our hyper-parameters in KGE are setup followed AMD [14].

The threshold for textual matching is 0.925, and the threshold for candidate selection is 0.9.

2.4. Datasets

We use the datasets provide by OAEI. AMD is able to be executed by organizers in four schema matching tracks, including Conference, Anatomy, bio-ML and Common Knowledge Graph track. However, AMD supports most of tracks in our local environment setups exclude for interactive matching track.

[1]https://github.com/pwin/owlready2
[2]https://huggingface.co/roberta-base

Table 1
Results of AMD for Anatomy and Conference tracks.

Track	runtime(s)	Precision	Recall	F1-score
Anatomy	160	0.953	0.817	0.88
Conference	-	0.82	0.41	0.55

2.5. Adaptations made for the evaluation

Our framework uses Python with Pytorch [3] and RDFLib [4], and is packed for SEALS using MELT. We use the best parameter set in local alignments for the OAEI submission, see section 2.3.

3. Results

3.1. Anatomy

The Anatomy track results of AMD are shown in Table 1. In this year, AMD returns 1299 correspondences in 160 seconds. The result shows that AMD can be competitive among the top promising matching systems. The mapping candidates generation runtime is still 3 seconds which is same as last year. However, to improve overall performance in terms of recall and F1-score, we conduct a filtering process with more time consuming.

3.2. Conference

The Conference track results of AMD are shown in Table 1. As expected, the performance of AMD in the conference track is not good, with the F-measure only slightly higher when comparing baseline method(StringEquiv). AMD shows a lack of ability to extract and match the properties in M2 and M3 evaluation variants. However, AMD has higher values in term of Precision in most tasks. We have 0.01 improvement in term of F1-score in this year.

3.3. bio-ML

Table 2 shows results of AMD in the bio-ML track. In this year, AMD is able to execute for bio-ML tasks on large ontologies. We use masked languange modeling in section 2.1 in semi-supervised tasks. We obtain promising and competitive results for most of the tasks in this track comparing with AMD performance in last year[14].

4. General comments

4.1. Comments on the result

Overall, the results show that AMD is able to complete several tasks in different domains on class-level matching in a timely manner. In this year, we have improvements in anatomy and

[3]https://pytorch.org
[4]https://github.com/RDFLib

Table 2
Results of AMD in bio-ML track.

Task	Unsupervised(90%)			Semi-supervised(70%)		
	Precision	Recall	F1-score	Precision	Recall	F1-score
OMIM-ORDO(Disease)	0.664	0.565	0.611	0.601	0.567	0.583
NCIT-DOID (Disease)	0.885	0.768	0.823	0.858	0.770	0.811
SNOMED-FMA (Body)	0.890	0.704	0.786	0.861	0.709	0.778
SNOMED-NCIT (Pharm)	0.962	0.745	0.840	0.952	0.746	0.836
SNOMED-NCIT (Neoplas)	0.836	0.534	0.652	0.792	0.528	0.633

bio-ML tracks in terms of evaluation metrics. By contrast to last year, we solved memory issues for large scale ontologies. Moreover, we consider to enable semi-supervised capability of AMD, and it is beneficial to train on triples in intra-ontology and inter-ontology.

However, AMD is still under development that it is only able to return class correspondences, and is not able to match properties and instances in the current stage for some tracks.

4.2. Improvements

The current development of AMD touches on several aspects. Besides considering properties and instances matching, we will utilize joint embedding to combine contextualized knowledge graph embeddings like coKE and additional knowledge resources such as WebIsA [21] as a lexicon database. Moreover, we will adapt AMD with more different data types parsing and parameters selections for different tracks.

5. Conclusions

In this paper, we have introduced an ontology matching system called AMD. In this year, we consider to use BERT-like pre-train language model to obtain contextualized representations. To improve the overall performance, we adapted a modified transR model to fit the ontology matching problem: thus, we learn low-dimensional representations for each class and relation to capture the hidden semantics of ontologies, rather than measuring the similarities between classes directly, as in other traditional systems. AMD makes full use of the textual and structure knowledge of ontologies. The results demonstrate the high efficiency and the promising performance of our proposed matching method as compared to other systems results in several tracks.

References

[1] I. F. Cruz, F. Palandri Antonelli, C. Stroe, AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies, PVLDB 2 (2009) 1586–1589.

[2] I. F. Cruz, F. Palandri Antonelli, C. Stroe, Efficient Selection of Mappings and Automatic Quality-driven Combination of Matching Methods, volume 551 of *CEUR Workshop Proceedings*, 2009, pp. 49–60.

[3] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, F. M. Couto, The AgreementMakerLight Ontology Matching System, in: International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE), Springer, 2013, pp. 527–541.
[4] Z. Sun, W. Hu, Q. Zhang, Y. Qu, Bootstrapping Entity Alignment with Knowledge Graph Embedding, in: IJCAI, volume 18, 2018, pp. 4396–4402.
[5] J. Euzenat, P. Shvaiko, Ontology Matching, Springer-Verlag, Heidelberg (DE), 2007.
[6] W. Li, X. Duan, M. Wang, X. Zhang, G. Qi, Multi-view embedding for biomedical ontology matching., OM@ ISWC 2536 (2019) 13–24.
[7] A. Laadhar, F. Ghozzi, I. Megdiche, F. Ravat, O. Teste, F. Gargouri, Partitioning and Local Matching Learning of Large Biomedical Ontologies, in: ACM SIGAPP Symposium on Applied Computing, 2019, pp. 2285–2292.
[8] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching, 2002, pp. 117–128.
[9] I. F. Cruz, W. Sunna, Structural Alignment Methods with Applications to Geospatial Ontologies, Transactions in GIS, Special Issue on Semantic Similarity Measurement and Geospatial Applications 12 (2008) 683–711.
[10] 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), 2018, pp. 787–798.
[11] P. Ristoski, J. Rosati, T. Di Noia, R. De Leone, H. Paulheim, RDF2Vec: RDF Graph Embeddings Their Applications, Semantic Web 10 (2019) 721–752.
[12] M. Cheatham, P. Hitzler, String similarity metrics for ontology alignment, in: International semantic web conference, Springer, 2013, pp. 294–309.
[13] J. Hao, M. Chen, W. Yu, Y. Sun, W. Wang, Universal Representation Learning of Knowledge Bases by Jointly Embedding Instances and Ontological Concepts, in: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1709–1719.
[14] Z. Wang, I. F. Cruz, Agreementmakerdeep results for oaei 2021., in: ISWC International Workshop on Ontology Matching (OM), CEUR Workshop Proceedings, CEUR-WS.org, 2021, pp. 124–130.
[15] J.-B. Lamy, Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies, Artificial intelligence in medicine 80 (2017) 11–28.
[16] L. Richardson, Beautiful soup documentation, April (2007).
[17] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019. URL: http://arxiv.org/abs/1908.10084.
[18] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: AAAI Conference on Artificial Intelligence, 2015.
[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.
[20] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint

arXiv:1907.11692 (2019).

[21] J. Seitner, C. Bizer, K. Eckert, S. Faralli, R. Meusel, H. Paulheim, S. P. Ponzetto, A large database of hypernymy relations extracted from the web., in: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), 2016, pp. 360–367.