SEBMatcher Results for OAEI 2022*

Francis Gosselin[1], Amal Zouaq[1]

[1]LAMA-WeST Lab, Departement of Computer Engineering and Software Engineering, Polytechnique Montreal, 2500 Chem. de Polytechnique, Montréal, QC H3T 1J4, Canada

Abstract

This paper presents the results of the Structural Embeddings with BERT Matcher (SEBMatcher) in the OAEI 2022 competition. SEBMatcher is a novel schema matching system that employs a 2 step approach: An unsupervised pretraining of a Masked Language Modeling BERT with random walks followed by a supervised training of a BERT for sequence classification with positive and negative mappings. This is the first year of participation in the OAEI for SEBMatcher and it has obtained promising results in participating tracks.

Keywords

Ontology alignment, Schema matching, Representation Learning, ISWC-2022

1. Presentation of the system

1.1. State, purpose, general statement

BERT [3] is a powerful state-of-the-art language representation model. The self-attention mechanism, at the core of its architecture, has been used in multiple tasks, including in ontology matching. Previous attention-based models for schema matching [9, 7, 4, 8] have obtained successful results, with BERT-like architectures such as BERTMap [4], DAEOM [9], or Finetom [8], graph attention [9] or customized attention layers [7]. SEBMatcher (see figure 1) is a system that proposes a new way to represent concepts by embedding structural information and neighbouring concepts through random walks. SEBMatcher realizes this in a 2-step approach. Firstly, it performs an unsupervised training of a BERT Masked Language Modeling (MLM) model, with the intention of reorganizing concepts in the embedding space with respect to the ontological structure. Secondly, the pre-trained BERT model (from the first step) is fine-tuned with an alignment classification task using positive and negative alignments that are obtained in various ways. For the OAEI evaluation, SEBMatcher was packaged using the MELT Web-packaging framework [5] .

Figure 1 presents the overall architecture of SEBMatcher at inference time, while Figure 2 describes SEBMatcher at training time. Note that the architectures during training and inference differ since some modules like candidate selection or greedy matcher do not need to be trained. In the following sections, we detail each component of these architectures.

Ontology Alignment Evaluation Initiative - OAEI 2022 Campaign, October 23rd, 2022, Hangzhou, China

*Corresponding author.

francis.gosselin@polymtl.ca (F. Gosselin); amal.zouaq@polymtl.ca (A. Zouaq)

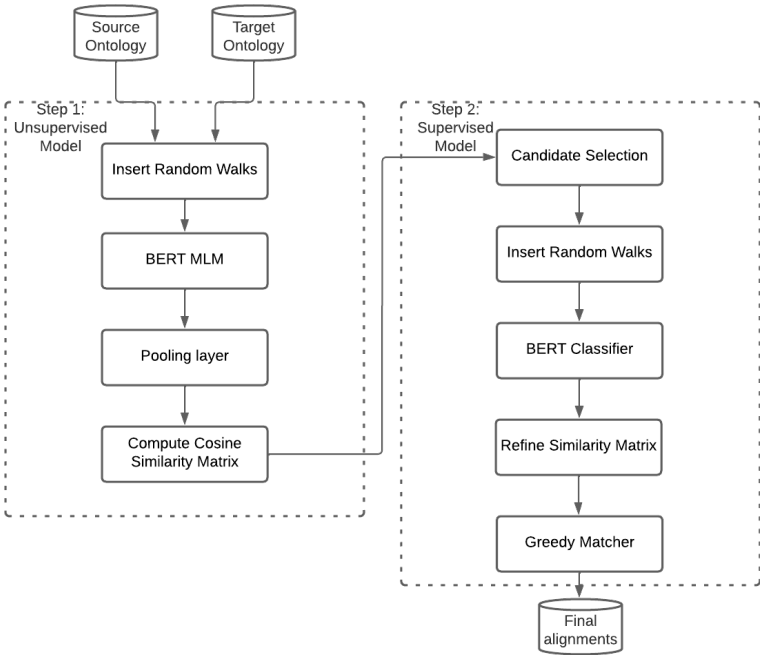

Figure 1: Architecture at inference. Note that preprocessing is done when loading the ontologies

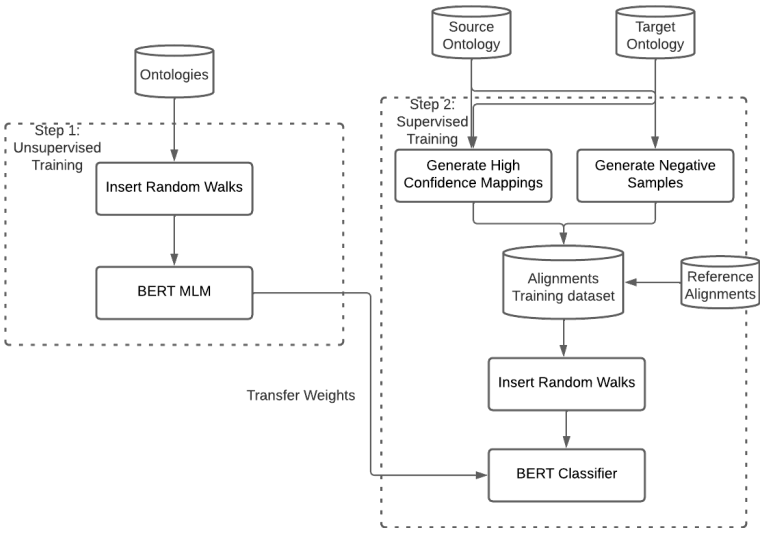

Figure 2: Architecture at training. Note that preprocessing is done when loading the ontologies

1.2. Specific techniques used

1.2.1. Preprocessing

During the preprocessing of the ontologies, we transform class and properties into tokens that can be processed by BERT. We apply a basic grammar correction for incorrect labels using

the python library autocorrect, an algorithm based on edit distance. We then add acronym resolution for niche acronyms and synonym resolution with Wordnet since not all ontologies define synonyms for their classes (oboInOwl:hasRelatedSynonym for example). We also apply basic preprocessing such as case folding, tokenization, and punctuation removal. Finally, we remove tokens that are not part of the BERT vocabulary.

1.2.2. Step 1: Unsupervised Model for Synonym and Context Embedding

The motive of this pre-training is firstly to introduce BERT to ontologies represented as sentences. But more importantly, it is to make a first refinement of the concept (class) embeddings, by making a concept and its synonyms close to each other in the vector space. This is done by using BERT's Masked language Modeling (MLM) training algorithm. The pre-training is crucial for the model since it is needed to understand information about the context and structure of an ontology. Our experiences have shown that without this pre-training, the model struggles to achieve a high score.

Random walks. Random walks [6] are how we have chosen to represent a concept and its context. In this step, a concept is replaced by its tree walk before being fed into BERT. The tree walk is essentially a set of random walks, defined in (algorithm 1), over the taxonomical structure and the object properties related to the concept. This algorithm takes as input a root concept from an ontology and outputs multiple random walks starting from the root concept. The resulting tree walk will not contain the same concept twice (aside from the root concept), this is done to add more diversity to the context of the root concept. If the ontology contains synonyms or multiple labels for the same concept, we randomly select one label every time we append this concept to a walk. To represent the relation SUBCLASS OF, the symbol < or > is used. If a child class is to the right of its parent class, we denote the relation as > e.g. "Person > Teacher", otherwise we use <.

Algorithm 1 Tree walk

Input: *Source ontology* O, *concept* c_0 *from the ontology* O
Output: *The Tree walk* W

Initialize set of visited nodes $C_v := \{\}$
Initialize walk $W := [\ ]$
$n\_branch \leftarrow randint(1..number\ of\ neighbours)$
for $i := 0$ *to* n_branch **do**
    $walk\_len \leftarrow randint(1..max\_len)$
    $Append\ c_0\ to\ W$
    for $j := 0$ *to walk* **do**
        $c_i \leftarrow last\_element(W)$
        $neighbours \leftarrow get\_neighbours(c_i)\ / C_v$
        $c_{i+1},\ relation \leftarrow choose\_random\_neighbour(neighbours)$
        Append $relation,\ c_{i+1}$ *to* W
        $C_v \leftarrow C_v \cup c_{i+1}$
    $Append\ END\_TOKEN\ to\ W$

Preprocessing step	Sample of data
Concept	ProgramCommitteeChair
Raw Tree Walk	ProgramCommitteeChair < ProgramCommitteeMember < Person
Tokenized Tree Walk	Program Committee Chair < Program Committee Member < Person
Masked Tree Walk	Program Committee Chair < [MASK] [MASK] [MASK] < Person

Table 1
Examples of Tree Walk's masking strategy

BERT Masked Language Modeling (MLM). The Masked Language Modeling learning technique consists of masking a certain portion of the input tokens (in our case tokens of concepts) and then trying to correctly predict the masked tokens. At inference, we omit the language modeling head since it is not needed. To preserve the lexical integrity of concepts, masks are applied to all their sub tokens. This rule does not apply to concepts with many tokens (more than 5) since the task would be too hard. 15% of concepts are masked. The pre-trained model ClinicalBERT [1] is used as initial weights for the model and was chosen for the Anatomy track. Given that the same model must be used across tracks in the OAEI competition, we kept the same model. However, more domain-related models could enhance the performance in specialized tracks.

Pooling layer. To retrieve the contextual embedding of a concept, its tree walk is firstly passed in the trained model. Then BERT's last hidden layer's output is filtered to only keep the tokens' embeddings corresponding to the root node. Finally, a MEAN pooling is applied to the filtered tokens' embeddings. For example, given a root node "Science" and the tree walk "Science < Subject; Science > Computer Science", we obtain Science's embedding by computing the average of the two "Science" tokens. This filtering is applied to obtain embeddings that do not drift too far from the concept, as it is hypothesized that the other concepts of the walk are not needed in the pooling layer since they already contextualize the concept with attention layers. This results in having a tree walk embedding of the root node that is not too far from its single concept embedding.

Similarity Matrix. To determine which candidate mappings (pairs of concepts) will be considered during inference, we perform a first computation of similarity based on the concept embeddings obtained after the pooling layer. Then we compute a matrix whose rows represent the concepts of the source ontology and whose columns represent the concepts of the target ontology, and for each combination, we return the cosine distance between their embeddings, as illustrated by formula 1.

$$S_{i,j} = cosine\_similarity(\Omega(c_i), \Omega(c'_j)) \tag{1}$$

Ω Function that transforms a concept into its tree walk embedding
c_i, c'_j the source and target concept respectively

1.2.3. Step 2: Supervised Alignment Scoring

The second step of the SEBMatcher system is a supervised classification task whose objective is to distinguish between positive and negative mappings or alignments. During this step, a BERT

for sequence classification takes pairs of tree walks (representing our concepts) as input and returns whether they are valid mappings or not. At inference time, candidate mappings are evaluated and only a subset is retained as valid candidates in the final alignments, including highly similar lexical-based alignments (string alignments).

Candidate Selection. Since the BERT classifier takes tree walk pairs and outputs a mapping score, it does not produce tree walks embedding. Thus we cannot directly compute a cosine similarity matrix from a dot product. In the supervised model, each mapping must be passed to the BERT classifier to obtain a similarity score. Given a source ontology O and a target ontology O', the computation of a similarity matrix of $|O|x|O'|$ possible alignments would require a vast amount of time. A conventional solution to this problem [4] is candidate selection, a module where we refine the scope of the calculations to highly probable matches. When comparing methods of obtaining candidate mappings, the best candidates should firstly have the highest Hits@K, where K is the number of the most probable target concepts to match for each source concept, and secondly should have a low computation time.

Our system's candidate selection is a mix of lexical matching and concept embedding similarities. Firstly, SEBMatcher computes string alignments. Those are mappings where the source and target concepts have at least one completely matching label or synonym. For optimization purposes, the string alignments are directly treated as part of the final alignments and are not passed to the BERT alignment classifier. Then, for each source concept, we take a subset of the |O'| possible mappings that consists of the $k = 10$ highest cosine similarity mappings, effectively reducing the number of considered candidates to $|O|xk$.

Alignments Training Dataset. The supervised model requires a dataset of negative and positive alignments during training. This dataset consists of reference alignments, string alignments and generated positive and negative alignments as described below:

- **Negative alignments** consist of a mix of hard and soft samples. Hard negative samples are negative alignments where the source and target concepts appear to be closely related. These samples are useful since they are important to correctly predict at inference time. Soft negative samples are alignments that the model should have no trouble ruling out since they are randomly chosen. The methods to generate these alignments are:
    1. hard intra-negative sampling: a negative alignment consisting of a random concept and one of its neighbour.
    2. hard inter-negative sampling: a negative sample made by taking a reference or string alignment and replacing either the source or target concept with its neighbouring concept.
    3. soft inter-negative sampling: a set of randomly chosen pairs of concepts from 2 different ontologies that assured to not be positive alignments.
- **Positive alignments** are a mix of reference alignments and generated alignments. The goal here is to have the most diverse set of alignments possible in order to regularize our model, the same way data augmentation is used in Machine Learning.
    1. reference mappings: 20% of the reference alignments. As per the OAEI rules, the system must be general, and cannot be fine-tuned for a single task. Therefore alignments from all tracks (Anatomy and Conference) where SEBMatcher participated were used.

2. string alignments: mappings computed in the candidate selection step that are considered positive alignments.
3. intra-positive sampling: a random concept paired with a copy of itself.

BERT Alignment Classifier. The scoring of a source and target tree walk pair is done with a BERT for sequence classification model that is fine-tuned from the initial weights of the BERT MLM model, the intuition behind this is that the classifier should have an easier process of learning if it benefits from the learned representations and attention weights during the MLM training. To produce a mapping score, the tree walks pair of the source and target concept are concatenated into one string along with the [SEP] token between them. Then, this string is passed to the BERT classifier which outputs the probability of the pair being a positive alignment. During training, at the start of step 2 (Supervised training), the weights of the BERT MLM are copied into the BERT classifier. Note that this cannot be done for the heads of the models since this is where both architectures differ. The weights of the BERT classifier are updated during training, but the BERT MLM model does not inherit these new weights.

Similarity Matrix Refinement. Now that each candidate has been given a mapping score by the BERT classifier, we are able to create a more accurate similarity matrix. The new similarity matrix contains the scores of all candidate mappings, while the score of each string alignment that was pruned during candidate selection is set to 1. All other possible mappings are set to zero.

Greedy Matcher. To select which mappings from the refined similarity matrix are going to be kept, we employ a greedy algorithm similar to other works like [7, 2]. This very simple algorithm iterates through candidate mappings from highest to lowest scores and selects every mapping whose source and target concepts are not part of an already selected mapping. The initial set of mappings is the set of string alignments computed during candidate selection. Furthermore, only mappings with scores higher than 0.85 are considered.

1.3. Parameters

For the tree walks, the number of branches is randomly chosen between 3 and 5. The path length for each branch is randomly chosen between 2 and 5. The BERT MLM model was trained for 40 epochs with a learning rate of 1e-5 with the ADAM optimizer while the BERT Alignment classifier was trained for 50 epochs with the same configuration. Both models have been trained on a RTX a6000 48GB graphic card.

2. Results

Full results for all reference alignments are shown on Table 2.

2.1. Anatomy

The anatomy track consists of matching the Adult Mouse Anatomy (MA) and the NCI Thesaurus describing the Human Anatomy (NCI). SEBMatcher reached an f1-score of 0.908, with a precision of 0.945 and a recall of 0.874. Compared to this year‘s other systems, SEBMatcher ranked 2 out

Table 2
SEBMatcher results in the anatomy and conference track

Reference alignments	F1-score rank (out of 11)	F1-score	precision	recall
ra1-m1	3	0.69	0.84	0.59
ra1-m2	-	-	-	-
ra1-m3	3	0.63	0.84	0.5
ra2-m1	3	0.65	0.8	0.55
ra2-m2	-	-	-	-
ra2-m3	3	0.59	0.80	0.47
rar2-m1	3	0.66	0.79	0.57
rar2-m2	-	-	-	-
rar2-m3	3	0.6	0.79	0.47
anatomy	2	0.908	0.945	0.874

of 10 in terms of f1 score. The runtime however is where SEBMatcher performed the worst, with a total time of 35602 seconds. This runtime includes training time (35350 seconds) and inference (252 seconds).

2.2. Conference

The conference track consists of matching a collection of ontologies describing the domain of organising conferences. This track contains different reference alignments sets, the M1 alignments contain only classes, M2 alignments are only properties, and M3 is both classes and properties. Since SEBMatcher currently matches exclusively classes, it does not perform well on the M3 reference alignments.

3. General comments and Conclusion

Overall, SEBMatcher obtained a high performance on the anatomy track but less interesting results in the conference track. Unfortunately, we found a bug in our greedy algorithm at the very end of the execution phase. After fixing it, our F1-score reaches a new score of 0.75 for the ra1-M1 alignments and has a similar score for the anatomy track. This is however not part of the official competition results. SEBMatcher is a system that fully exploits the benefits of rich ontologies, since it can more easily organize elements in the embedding space when there is a well-defined ontological structure. Thus flat ontologies would not be usable with our architecture.

There is still a lot of room for improvement. Firstly, a way to improve performance would be to generate more semantically complex positive alignments, this refers to the fact that string alignments are easy mappings to find and the system benefits a lot more by learning from difficult positive alignments. Secondly, other transformer architectures than BERT could be explored. SEBMatcher produces mapping probabilities that are often close to either 1 or 0 and are rarely far away from both values. This is undesirable for filtering valid candidates.

It is also to be noted that SEBMatcher withdrew from the BioML track since we did not have enough time to adapt the system to large ontologies. In fact, this brings up a main downside of SEBMatcher which is its runtime. This is due to our BERT pre-training and fine-tuning. At inference, however, SEBMatcher performs reasonably. In our future work, we plan to condense these two training steps into one process. Another way to improve runtime would be to prioritize concepts that are likely to be matched and ignore those that are not. We plan to continue the exploration of concept embeddings and the generalization of our approach to other ontologies and tracks.

4. Acknowledgements

This research has been funded by Canada's NSERC Discovery Research Program.

References

[1] Emily Alsentzer et al. *Publicly Available Clinical BERT Embeddings*. 2019. DOI: 10.48550/ARXIV.1904.03323. URL: https://arxiv.org/abs/1904.03323.

[2] Alexandre Bento, Amal Zouaq, and Michel Gagnon. "Ontology Matching Using Convolutional Neural Networks". English. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 5648–5653. ISBN: 979-10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.693.

[3] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[4] Yuan He et al. "Bertmap: A bert-based ontology alignment system". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 5. 2022, pp. 5684–5691.

[5] Sven Hertling, Jan Portisch, and Heiko Paulheim. "MELT - Matching EvaLuation Toolkit". In: *Semantic Systems. The Power of AI and Knowledge Graphs - 15th International Conference, SEMANTiCS 2019, Karlsruhe, Germany, September 9-12, 2019, Proceedings*. 2019, pp. 231–245. DOI: 10.1007/978-3-030-33220-4_17. URL: https://doi.org/10.1007/978-3-030-33220-4%5C_17.

[6] Ole Magnus Holter et al. "Embedding owl ontologies with owl2vec". In: *CEUR Workshop Proceedings*. Vol. 2456. Technical University of Aachen. 2019, pp. 33–36.

[7] Vivek Iyer, Arvind Agarwal, and Harshit Kumar. "Multifaceted Context Representation using Dual Attention for Ontology Alignment". In: *CoRR* abs/2010.11721 (2020). arXiv: 2010.11721. URL: https://arxiv.org/abs/2010.11721.

[8] Leon Knorr and Jan Portisch. "Fine-TOM matcher results for OAEI 2021". In: *CEUR Workshop Proceedings*. Vol. 3063. RWTH. 2022, pp. 144–151.

[9] Jifang Wu et al. "Daeom: A deep attentional embedding approach for biomedical ontology matching". In: *Applied Sciences* 10.21 (2020), p. 7909.