TOM Matcher Results for OAEI 2021

Daniel Kossack[1][0000−0002−8649−3357], Niklas Borg[1][0000−0002−8081−6653], Leon Knorr[1][0000−0003−4117−2629], and Jan Portisch[1,2][0000−0001−5420−0663]

[1] SAP SE, Walldorf, Germany
{daniel.tobias.kossack, niklas.borg, leon.knorr, jan.portisch}@sap.com
[2] Data and Web Science Group, University of Mannheim, Germany
jan@informatik.uni-mannheim.de

Abstract. This paper presents the matching system TOM together with its results in the Ontology Alignment Evaluation Initiative 2021 (OAEI 2021). This is the first participation of TOM in the OAEI.
Very recently, transformers achieved remarkable results in the natural language processing community on a variety of tasks. The TOM matching system exploits a zero-shot transformer-based language model to calculate confidences for each instance. The matcher uses the pre-trained transformer model *paraphrase-TinyBERT-L6-v2*.[3]

Keywords: Ontology Matching · Ontology Alignment · Language Models · Transformers

1 Presentation of the System

1.1 State, purpose, general statement

Transformers for Ontology Matching (TOM) is a matching system that uses a transformer-based language model to calculate a confidence for a pair of entities. The matcher is implemented as a pipeline of subsequent steps using pre-defined matching modules of the *Matching EvaLuation Toolkit (MELT)* [6], a framework for ontology matching and evaluation. Particularly, the new transformer extension of MELT [7] is used in the implementation of this matcher. The matcher was implemented and packaged as a Docker image implementing the new *Web Interface*[4].

1.2 Specific Techniques Used

Transformer-based language models Transformers are types of artificial neural networks that have a specific architecture [13]. Especially in the domain of natural language processing, transformers achieved good results on a variety

[3] Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[4] https://dwslab.github.io/melt/matcher-packaging/web#web-interface-http-matching-interface

of tasks [13,4]. The tasks a transformer is capable of carrying out depend on its architecture and the task it was trained on.

To perform well, transformers need to be trained on large amounts of data. Many researchers and organizations train their transformers on textual data from various domains. After training the model (so called *pre-training*), the transformer models can be *fine-tuned* for specific tasks or domains. Typically, the *fine-tuning* process is computationally cheap compared to training the main model. However, training data is required.

The TOM research project explored the capabilities of such pre-trained models for ontology matching. TOM does not use fine-tuned models, but there is a fine-tuned version of TOM, which is named *Fine-TOM* (F-TOM) [9].

There are several python libraries, like the transformers library by huggingface[5] [14] and the sentence-transformers library by the Ubiquitous Knowledge Processing Lab[6] [12] that provide access to a variety of pre-trained transformers. We evaluated the performance of several pre-trained models, inter alia, GPT-2 [11], BERT [4], and different Sentence-BERT models [12] on the Anatomy, Conference, and Knowledge Graph track. Based on this evaluation, we decided to use the Sentence-BERT model *paraphrase-TinyBERT-L6-v2* [12] for our submitted matcher since it achieved the best results.

TOM Pipeline TOM consists of five components that are arranged in a pipeline which is shown in Figure 1. The arrows indicate how the ontologies and alignments are passed between the components. Each component can be conceptually regarded as a matcher. This design pattern is proposed by MELT. For chaining the components, class `MatcherYAAAJena` was used.

First, the ontologies are aligned in with string matching methods. Since these are typically of high precision, the resulting alignment of this step is directly added to the final alignment.
The transformer component in MELT is, by default, implemented as a filter. Therefore, candidates have to be generated which are then filtered by the transformer. In this case, a string overlap metric is used. The transformer component adds a confidence to each candidate. After the transformer matcher calculated the confidence for each candidate pair, the alignment is filtered by a threshold. Based on an evaluation on different OAEI tracks, we decided to set the threshold to 0.8. Since most OAEI datasets are typically of one-to-one parity, we use an efficient implementation of the Hungarian method, known as *Maximum Weight Bipartite Matching* (MWBM) [3]. The motivation behind the matching components and their details are described in the sections below.

String Matcher The string matcher is used to find very obvious correspondences. To do that it compares the inputs word by word. It assigns a confidence of 1.0 to the obvious correspondences. Therefore, the resource-consuming transformer matcher does not have to evaluate those pairs of elements again. We

[5] https://huggingface.co
[6] https://www.sbert.net

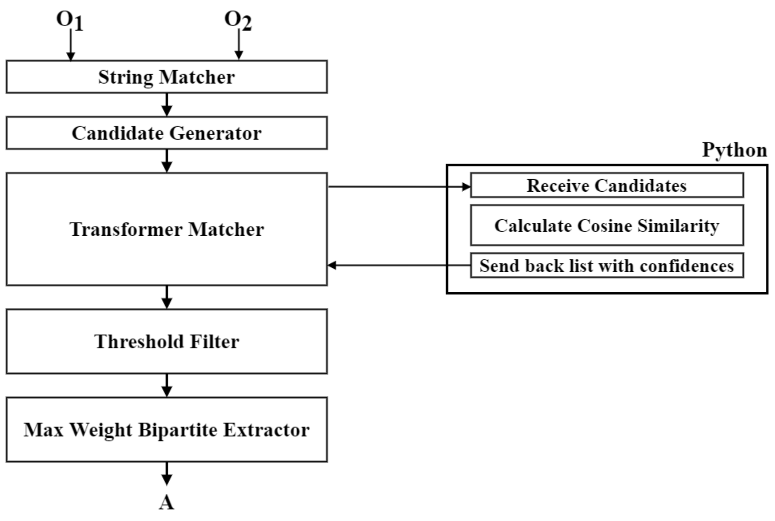

Fig. 1. High-level view of the TOM matching process. O_1 and O_2 represent the input ontologies and optionally elements. The final alignment is referred to as A.

use the default string matcher implementation of MELT (class `SimpleString-Matcher`).

Candidate Generator The Candidate Generator is used to generate an input alignment for the transformer matcher. It creates a cross product of both ontologies, so that the alignment consists of all possible pairs of elements. For large ontologies, the alignment would be too large to be proceeded by the transformer matcher within an appropriate time frame. Therefore, the Candidate Generator excludes the correspondences that are found by the string matcher and obvious non-correspondences.

Those are found through broad string operations. The candidate generator class splits the labels and saves single words in a set. Then the sets are compared and if over 50 percent of the words are equal, the correspondence is further processed by the transformer matcher. If less than 50 percent of the words are equal, it is not necessary to calculate a similarity for this pair. Finally, the alignment is passed to the transformer matcher.

Transformer Matcher The transformer matcher iterates over the alignment and passes each pair of elements to the sentence-bert library [12]. For each pair, it receives a similarity value $s \in [0,1]$ which is used as the assigned confidence that this pair is a match. The transformer matcher does not change the size of the alignment but only adds additional confidences. To access a transformer via the sentence-BERT library, the transformer matcher starts a python server in the background since the transformer libraries are implemented in Python while the matching pipeline is implemented in Java. The communication between the Java project and the python server works via an HTTP Application Programming Interface (API) that is represented by the horizontal arrows in Figure 1. After the python server received the pairs of elements and calculated the similarity values with the cosine similarity function of scikit-learn[7] [10], it returns the confidence list back to the class `Transformer Matcher`. Then, the transformer matcher replaces the confidences in the current alignment.

Threshold Filter The threshold filter uses the alignment and cuts off all correspondences with a low confidence. To do that, the filter uses a threshold between zero an one. All correspondences with a confidence lower than the threshold are excluded from the alignment. The submitted matcher has a threshold $t = 0.8$ which yielded good results for all evaluated tracks. We use the default string matcher implementation of MELT (class `ThresholdFilter`).

Max Weight Bipartite Extractor Up to this step in the pipeline, we could have multiple correspondences for an ontological element in the alignment. Therefore, the max weight bipartite extractor converts the current state to a one-to-one alignment. We use the default class `MaxWeightBipartiteExtractor` of MELT.

2 Results

This section discusses the results of TOM for the tracks of OAEI 2021 on which the matcher was able to produce results. These include the Anatomy [1], Conference [2,15], and Knowledge Graph track [8,5].

2.1 Anatomy Track

TOM could achieve a higher F-measure than the OAEI Baseline (0.866 vs. 0.766). It was noticeable that the recall was improved by TOM (0.808 vs 0.622) but the precision is lower than with only the OAEI Baseline (0.997 vs 0.933). So there are non-obvious matches that cannot be found with string based matching but by TOM. Examples for these would be the matches *Parietal Lobe of the Brain* & *parietal cortex* with a confidence of 0.8202 and *great vein of heart* & *Great Cardiac Vein* with a confidence of 0.8794. Those are found because of the ability of transformers to detect semantic similarities between two phrases or words such as *heart* and *cardiac*, even though they are spelled differently.

2.2 Conference Track

Also on the Conference track, TOM is able to find more correspondences than the OAEI Baseline. So the recall is higher (0.48 vs 0.41) and also the F-measure is higher (0.57 vs 0.53). The precision is a bit lower (0.69 vs 0.76).

[7] `https://github.com/scikit-learn/scikit-learn/blob/844b4be24/sklearn/metrics/pairwise.py#L1211l`

3 General Comments

We thank the OAEI organizers for their support and commitment.

4 Conclusion

In this paper, we presented the TOM matching system and its results in the OAEI 2021. We can conclude that transformer-based language models are able to improve performance in the task and process of ontology matching.

The developed matching system achieves an overall better F-measure than the baseline matchers and it improves the recall. It is important to note that the highest possible recall is set by the Candidate Generator's alignment. The Transformer Matcher works only with this alignment and so it is not possible to achieve a higher recall.

The research also showed that the presented architecture and the implementation in Java and Python are appropriate approaches to use transformers for ontology matching. Most of the used matching components are available via the MELT framework to allow other developers to re-use them. The docker packaging allows to submit any implementation without set-up efforts on the organizer side which is also beneficial for matcher developers who do not have to worry about the execution of their system.

This is the first OAEI participation of TOM and the system can be greatly improved in the future, for example by using fine-tuned models or by improving the candidate generation pipeline. The reported results motivate further research in the area of transformer-based ontology matching.

References

1. Bodenreider, O., Hayamizu, T.F., Ringwald, M., de Coronado, S., Zhang, S.: Of mice and men: Aligning mouse and human anatomies. In: AMIA 2005, American Medical Informatics Association Annual Symposium, Washington, DC, USA, October 22-26, 2005. AMIA (2005), https://knowledge.amia.org/amia-55142-a2005a-1.613296/t-001-1.616182/f-001-1.616183/a-012-1.616655/a-013-1.616652
2. Cheatham, M., Hitzler, P.: Conference v2.0: An uncertain version of the OAEI conference benchmark. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandecic, D., Groth, P., Noy, N.F., Janowicz, K., Goble, C.A. (eds.) The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II. Lecture Notes in Computer Science, vol. 8797, pp. 33–48. Springer (2014). https://doi.org/10.1007/978-3-319-11915-1_3, https://doi.org/10.1007/978-3-319-11915-1_3
3. Cruz, I.F., Antonelli, F.P., Stroe, C.: Efficient selection of mappings and automatic quality-driven combination of matching methods. In: Shvaiko, P., Euzenat, J., Giunchiglia, F., Stuckenschmidt, H., Noy, N.F., Rosenthal, A. (eds.) Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) collocated with the 8th International Semantic Web Conference (ISWC-2009) Chantilly, USA,

October 25, 2009. CEUR Workshop Proceedings, vol. 551. CEUR-WS.org (2009), http://ceur-ws.org/Vol-551/om2009_Tpaper5.pdf

4. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805
5. Hertling, S., Paulheim, H.: The knowledge graph track at OAEI - gold standards, baselines, and the golden hammer bias. In: Harth, A., Kirrane, S., Ngomo, A.N., Paulheim, H., Rula, A., Gentile, A.L., Haase, P., Cochez, M. (eds.) The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12123, pp. 343–359. Springer (2020). https://doi.org/10.1007/978-3-030-49461-2_20, https://doi.org/10.1007/978-3-030-49461-2_20
6. Hertling, S., Portisch, J., Paulheim, H.: MELT - matching evaluation toolkit. In: Acosta, M., Cudré-Mauroux, P., Maleshkova, M., Pellegrini, T., Sack, H., Sure-Vetter, Y. (eds.) Semantic Systems. The Power of AI and Knowledge Graphs - 15th International Conference, SEMANTiCS 2019, Karlsruhe, Germany, September 9-12, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11702, pp. 231–245. Springer (2019). https://doi.org/10.1007/978-3-030-33220-4_17
7. Hertling, S., Portisch, J., Paulheim, H.: Matching with transformers in MELT. CoRR **abs/2109.07401** (2021), https://arxiv.org/abs/2109.07401
8. Hofmann, A., Perchani, S., Portisch, J., Hertling, S., Paulheim, H.: Dbkwik: Towards knowledge graph creation from thousands of wikis. In: Nikitina, N., Song, D., Fokoue, A., Haase, P. (eds.) Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017. CEUR Workshop Proceedings, vol. 1963. CEUR-WS.org (2017), http://ceur-ws.org/Vol-1963/paper540.pdf
9. Knorr, L., Portisch, J.: Fine-TOM matcher results for OAEI 2021. In: OM@ISWC 2021 (2021), to appear
10. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
11. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2018), https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf
12. Reimers, N., Gurevych, I.: 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 (11 2019), http://arxiv.org/abs/1908.10084
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. CoRR **abs/1706.03762** (2017), http://arxiv.org/abs/1706.03762
14. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Brew, J.: Huggingface's transformers: State-of-the-art natural language processing. CoRR **abs/1910.03771** (2019), http://arxiv.org/abs/1910.03771
15. Zamazal, O., Svátek, V.: The ten-year ontofarm and its fertilization within the onto-sphere. J. Web Semant. **43**, 46–53 (2017). https://doi.org/10.1016/j.websem.2017.01.001, https://doi.org/10.1016/j.websem.2017.01.001