Ontology Matching using Textual Class Descriptions

Yiwen Peng[1], Mehwish Alam[1] and Thomas Bonald[1]

[1] *Télécom Paris, Institut Polytechnique de Paris, France*

Abstract

In this paper, we propose TEXTO, a **TEXT**-based **O**ntology matching system. This matcher leverages the rich semantic information of classes available in most ontologies by a combination of a pre-trained word embedding model and a pre-trained language model. Its performance is evaluated on the datasets of the OAEI Common Knowledge Graphs Track, augmented with the description of each class, and a new dataset based on the refreshed alignment of Schema.org and Wikidata. Our results demonstrate that TEXTO outperforms all state-of-art matchers in terms of precision, recall and F1 score. In particular, we show that almost perfect class alignment can be achieved using textual content only, excluding any structural information like the graph of classes or the instances of each class.

Keywords

Ontology Matching, Language Models, Textual Information

1. Introduction

Ontology matching [1] is the task of finding mappings between classes of two ontologies. The Ontology Alignment Evaluation Initiative (OAEI)[1] provides several benchmark datasets along with a unified evaluation methodology for this problem. Most datasets of the OAEI contain structural information, which is exploited by many matching systems [2]. In practice, this requires the ontologies to be well-structured, which is not necessarily the case for cross-domain and automatically generated Knowledge Graphs (KGs).

In this paper, we propose an approach called TEXTO for TEXT-based Ontology matching, leveraging the textual descriptions of the classes through a language model, and excluding any structural information. Our experiments show that TEXTO achieves nearly optimal alignment on the common KGs given in OAEI 2022, with an average increase of 13% in recall and 7% in F1 score over the state-of-the-art methods. TEXTO has also been tested on a new dataset based on the refreshed alignment between Schema.org and Wikidata, where the predicted matching is also almost perfect. The source code and the dataset are available online[2].

The paper is structured as follows. Section 2 presents the related work. Section 3 describes the TEXTO system, and Section 4 presents the datasets and the experimental results. Section 5 concludes our work.

OM 2023: The 18th International Workshop on Ontology Matching collocated with the 22nd International Semantic Web Conference ISWC-2023 November 7th, 2023, Athens, Greece

yiwen.peng@telecom-paris.fr (Y. Peng); mehwish.alam@telecom-paris.fr (M. Alam); thomas.bonald@telecom-paris.fr (T. Bonald)
0000-0002-7867-6612 (M. Alam)

[1] https://oaei.ontologymatching.org/
[2] https://github.com/peng-yiwen/Ontolgy_matching

2. Related Work

Ontology matching methods can be broadly classified into two categories: instance-based methods and schema-based methods. Instance-based methods leverage entity-level information, such as the classes and properties of each entity [3, 4, 5]. The general principle is to match classes sharing a similar set of instances. In contrast, schema-based approaches use the ontology schema only, like the labels of classes and the links between classes. For example, LsMatch uses string similarity and synonyms to match classes based on the labels [6], while Matcha exploits both lexical and structural information [7]. Logic-aware strategies, such as LogMap [8], use lexical indexing and disjointness-based reasoning. With the success of large language models, text-enhancement techniques have grown in popularity. For instance, MEDTO system [9] applies SBERT to initiate the vector representation for each node of a graph neural network for matching medical ontologies. AMD [10] applies a knowledge graph embedding technique for filtering the candidates generated by a language model. Another matcher [11], similar to ours, uses a pre-trained SBERT model for encoding labels. However, it was designed for the Anatomy and Conference Track of OAEI and does not leverage the description of the classes. Finally, a number of other methods exploit both the schema and the instances, the latter being typically used in the final step to filter candidates generated by the analysis of the schema [12, 13].

3. TEXT-based Ontology matching (TEXTO)

The overview of TEXTO is shown in Figure 1. The objective is to map two ontologies, whose classes are respectively indexed by *i* and *j*. Observe that the two ontologies might have different numbers of classes. The pipeline consists of five steps, as described below.

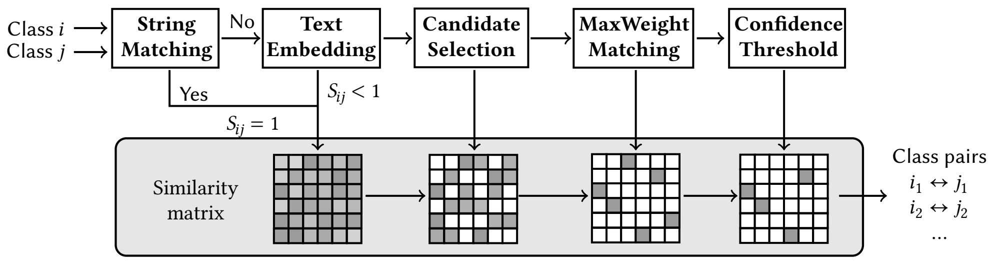

Figure 1: Overview of the TEXTO matching pipeline.

Step 1: String Matching. In the first step, the label of each class is preprocessed through tokenization (white space and camel case tokenization, splitting of multi-word expressions such as *televisionnetwork*) and stop word removal. After text preprocessing, any class pair i, j sharing exactly the same labels is assigned a similarity score of 1; the other classes are passed to Step 2.
Step 2: Text Embedding. In Step 2, a similarity score in $[-1, 1]$ is computed for each class pair i, j whose labels do not match exactly (cf. Step 1). First, a vector representation of the label of each class i, j is generated by the GloVe model[3], using the average of the vectors obtained for

[3] https://huggingface.co/fse/glove-wiki-gigaword-300

each word of the label (mean pooling). Second, a vector representation of the class description is obtained through Sentence-BERT (SBERT), a language model trained by contrastive learning for Semantic Textual Similarity tasks. Here, the pre-trained model *all-MPNet-base-v2*[4] is used, which provides the best quality among all SBERT models available on Hugging Face[5].

Denoting by $v_i^{\text{label}}, v_j^{\text{label}}$ the vector representations of the *labels* of classes i, j, and $v_i^{\text{desc}}, v_j^{\text{desc}}$ the vector representations of the *descriptions* of classes i, j, the similarity score between i and j is computed as the weighted sum:

$$S_{ij} = \alpha \cos(v_i^{\text{label}}, v_j^{\text{label}}) + (1 - \alpha) \cos(v_i^{\text{desc}}, v_j^{\text{desc}}) \tag{1}$$

where $\alpha \in [0, 1]$ is some hyper-parameter.

Step 3: Candidate Selection. In this step, top-k candidates are selected in terms of similarity. Specifically, a class pair i, j is selected as a candidate either if i ranks among the top-k classes for matching j or if j ranks among the top-k classes for matching i. The similarity scores of the selected candidates remain unchanged, while the others are set to 0 (see Figure 1).

Step 4: Max-Weight Matching. The new similarity matrix, after Step 3, is used to match the class pairs by Max-Weight Matching (MWM). Specifically, MWM finds a one-to-one mapping that maximizes the total similarity [14]. When the two ontologies have different numbers of classes, say n and m, MWM returns $\min(n, m)$ pairs.

Step 5: Confidence Threshold. After Step 4, some class pairs might have low similarity due to the one-to-one mapping constraint. To solve this, a confidence threshold t is applied so that any matched class pair i, j with a similarity score S_{ij} less than t is removed from the prediction.

4. Datasets and Evaluation

Datasets. TEXTO is evaluated on two benchmarks of the OAEI Common Knowledge Graphs Track[6]. The first benchmark **NELL-DBpedia** has been annotated by humans and verified by experts. Note that the alignment is partial, not every class of an ontology has an equivalent class in the other. The second benchmark **YAGO-Wikidata** was originally created from the mapping between Schema.org and Wikidata[7] used in YAGO [15].

The textual descriptions of the classes are absent in both datasets. For the NELL-DBpedia dataset, the missing descriptions of DBpedia classes are completed with those of the equivalent Wikidata classes. For the YAGO-Wikidata dataset, the descriptions are directly extracted from the corresponding databases, Schema.org and Wikidata.

In addition to these two reference datasets, a new dataset is proposed called **Schema-Wikidata**, based on the refreshed alignment of Schema.org and Wikidata. Note that the alignment proposed in the benchmark YAGO-Wikidata dates back to 2017. Some class URIs are deprecated. Our dataset is based on the SPARQL query on Wikidata using the *owl:equivalentClass* predicate (property P1706). The new dataset consists of 343 class pairs, 39 more than those available in the YAGO-Wikidata dataset. The new dataset is available in the OAEI

[4]https://huggingface.co/sentence-transformers/all-mpnet-base-v2
[5]https://www.sbert.net/docs/pretrained_models.html
[6]http://oaei.ontologymatching.org/2022/commonKG/index.html
[7]https://github.com/okfn-brasil/schemaOrg-Wikidata-Map

standard[8] on the GitHub repository of the paper. The statistics of the considered datasets are shown in Table 1.

Table 1
Statistics of the considered datasets (number of classes and gold mappings).

Dataset	#Classes	#Gold mappings
NELL / DBpedia	134 / 138	129
Wikidata / YAGO	304 / 304	304
Schema / Wikidata	343 / 343	343

Results. TEXTO is implemented in Python using Pytorch and Owlready2[9]. Our evaluation is completed in under 5 minutes on a Linux machine with 62 GB of RAM and 8 CPUs (3.47GHz) processors. For both the NELL-DBpedia and Schema-Wikidata datasets, the hyper-parameter α weighting label and description similarities is set to 0.5; for the YAGO-Wikidata dataset, it is set to 0.4. For all datasets, the parameter k for candidate selection is set to 5, and the confidence threshold t is set to 0.4. Table 2 reports the results of the experiments in terms of precision, recall and F1-score, for the two benchmarks of the OAEI, for which results of the state-of-the-art methods are available. For the dataset NELL-DBpedia, where the alignment is partial, any prediction involving both classes that are not mapped in the gold standard is ignored.

Table 2
Comparison of TEXTO with state-of-the-art methods

Method	**NELL-DBpedia**			**YAGO-Wikidata**		
	P	**R**	**F1**	**P**	**R**	**F1**
KGMatcher+ [4]	**1.00**	0.91	0.95	0.99	0.84	0.91
Matcha [7]	**1.00**	0.81	0.90	**1.00**	0.80	0.89
ATMatcher [12]	**1.00**	0.80	0.89	**1.00**	0.77	0.87
ALOD2Vec [13]	**1.00**	0.80	0.89	0.99	0.77	0.86
LsMatch [6]	0.96	0.75	0.84	0.96	0.63	0.76
LogMap [8]	0.99	0.80	0.88	**1.00**	0.76	0.86
TEXTO	**1.00**	**0.99**	**1.00**	**1.00**	**0.99**	**0.99**
	-	+9%	+5%	-	+18%	+9%

By incorporating extra information such as descriptions, TEXTO outperforms all state-of-the-art methods that only consider labels, achieving the highest precision, recall, and F1 scores.

Ablation study. We now present an ablation study of TEXTO on the three datasets, focusing on the processing of textual information and the impact of MWM. The results are shown in Table 3. For methods without MWM, the top-1 candidate with positive similarity is selected. The methods with MWM component also include candidate selection (Step 3) and confidence threshold (Step 5).

[8]https://moex.gitlabpages.inria.fr/alignapi/format.html
[9]https://owlready2.readthedocs.io/en/latest/

Table 3
Ablation Study. The following abbreviations are used to represent each step of our matcher: String Matching = String; Embedding = E, with E(label) for labels only, E(label, desc) for labels and descriptions, E*(label, desc) when SBERT is used for both labels and descriptions; Max-Weight Matching = MWM.

Method	NELL-DBpedia			YAGO-Wikidata			Schema-Wikidata		
	P	R	F1	P	R	F1	P	R	F1
String	**1.00**	0.78	0.88	**1.00**	0.71	0.83	**1.00**	0.67	0.80
String + E(label)	0.97	**0.99**	0.98	0.93	0.93	0.93	0.94	0.94	0.94
String + E(label, desc)	0.98	**0.99**	0.99	0.97	0.97	0.97	0.95	0.95	0.95
String + E(label, desc) + MWM	**1.00**	**0.99**	**1.00**	**1.00**	**0.99**	0.99	0.99	**0.98**	**0.99**
String + E*(label, desc)	0.98	**0.99**	0.99	0.97	0.97	0.97	0.96	0.96	0.96
String + E*(label, desc) + MWM	0.99	0.98	0.98	**1.00**	**0.99**	**1.00**	0.99	0.97	0.98

The ablation study reveals that leveraging the textual information contained in class descriptions increases both precision and recall, a trend prominently observed in the YAGO-Wikidata and Schema-Wikidata datasets. In the case of NELL-DBpedia, the addition of class descriptions does not improve recall. A potential explanation is the limited number of classes in this dataset, making pure label information sufficient for ontology matching. Finally, replacing GloVe by SBERT for embedding the labels tends to decrease performance, suggesting that a word embedding model is more appropriate than a language model for encoding labels.

Discussion. In the benchmarks used in the literature for common KGs, the set of available classes is restricted to the set of classes for which the gold mapping is available (or slightly larger, cf. Table 1). In reality, the number of classes to match is much higher (there are more than 2M classes in Wikidata, for instance) and these classes have a complex, hierarchical structure that makes matching much more difficult. This raises the following question:

What is the effectiveness of existing methods beyond the benchmarks of the literature?

To this end, further experiments have been conducted on expanded versions of the YAGO-Wikidata and Schema-Wikidata datasets. Specifically, we have added all 1,360 classes of Schema.org instead of only those for which the gold standard is known. These additional classes could be sub-classes, super-classes, or siblings of the original classes. The results of TEXTO on these extended datasets are presented in Table 4.

Table 4
Performance of TEXTO on extended versions of the datasets

Method	YAGO-Wikidata+			Schema-Wikidata+		
	P	R	F1	P	R	F1
TEXTO	0.94	0.94	0.94	0.96	0.95	0.96

We observe that TEXTO still achieves good performance in these more realistic scenarios, although the alignment is no longer perfect. This suggests that richer benchmarks should be considered for ontology matching, to make the task more challenging.

5. Conclusion

In this paper, we have introduced an ontology matching system called TEXTO, which leverages both the label and the textual description of each class. This matcher combines the vector representations obtained from the GloVe word embedding model for the label and from the SBERT language model for the description. The resulting similarity scores are then used to derive the best mapping.

Our system has been tested on the OAEI benchmarks, enriched with the description of each class, and on a new dataset matching Schema.org and Wikidata. The results show that TEXTO achieves almost perfect alignment, using textual information only. This calls for the publication of new benchmarks with more classes, most likely without complete alignment, to make the task more challenging. In such more realistic scenarios, the question of the respective importance of textual content and structural information for ontology matching remains open.

References

[1] J. Euzenat, P. Shvaiko, Ontology matching, 2nd ed., Springer-Verlag, Heidelberg (DE), 2013.

[2] O. Fallatah, Z. Zhang, F. Hopfgartner, A gold standard dataset for large knowledge graphs matching, in: OM@ISWC, 2020.

[3] H. Belhadi, K. Akli-Astouati, Y. Djenouri, J. C.-W. Lin, Data mining-based approach for ontology matching problem, Applied Intelligence (2020).

[4] O. Fallatah, Z. Zhang, F. Hopfgartner, The impact of imbalanced class distribution on knowledge graphs matching, in: OM@ISWC, 2022.

[5] D. Ayala, I. Hernández, D. Ruiz, E. Rahm, Towards the smart use of embedding and instance features for property matching, in: ICDE, 2021.

[6] A. Sharma, A. Patel, S. Jain, LSMatch and LSMatch-Multilingual results for OAEI, in: OM@ISWC, 2022.

[7] D. Faria, M. C. Silva, P. Cotovio, P. Eugénio, C. Pesquita, Matcha and Matcha-DL results for OAEI 2022, in: OM@ISWC, 2022.

[8] E. Jiménez-Ruiz, LogMap family participation in the OAEI 2022, in: OM@ISWC, 2022.

[9] J. Hao, C. Lei, V. Efthymiou, A. Quamar, F. Özcan, Y. Sun, W. Wang, Medto: Medical data to ontology matching using hybrid graph neural networks, in: KDD, 2021.

[10] Z. Wang, AMD results for OAEI 2022, in: OM@ISWC, 2022.

[11] D. Kossack, N. Borg, L. Knorr, J. Portisch, TOM matcher results for OAEI 2021, in: OM@ISWC, 2022.

[12] S. Hertling, H. Paulheim, ATBox results for OAEI 2022, in: OM@ISWC, 2022.

[13] J. Portisch, H. Paulheim, Alod2vec matcher results for OAEI 2021, in: OM@ISWC, 2021.

[14] R. Jonker, T. Volgenant, A shortest augmenting path algorithm for dense and sparse linear assignment problems, in: DGOR/NSOR, 1988.

[15] T. Pellissier Tanon, G. Weikum, F. Suchanek, Yago 4: A reason-able knowledge base, in: ESWC, 2020.