Biomedical Ontology Alignment with BERT

Yuan He[1], Jiaoyan Chen[1], Denvar Antonyrajah[2], and Ian Horrocks[1]

[1] Department of Computer Science, University of Oxford, UK
[2] Samsung Research, UK

Abstract. Existing machine learning-based ontology alignment systems often adopt complicated feature engineering or traditional non-contextual word embeddings. However, they are often outrun by the rule-based systems despite the model complexity. This paper proposes a novel ontology alignment system based on a contextual embedding model named BERT, aiming to sufficiently utilize the text semantics implied by ontologies. Our results on two biomedical alignment tasks demonstrate that, despite using the to-be-aligned classes alone as the input, our system outperforms the leading systems: LogMap and AML.

Keywords: Ontology Alignment · Contextual Embeddings · BERT.

1 Introduction

Ontology alignment refers to matching semantically related entities from different ontologies, with the vision of integrating data from heterogeneous resources. The resulting mappings usually indicate equivalence or subsumption relationships, consequently providing a convenient means for merging two ontologies. Moreover, through alignment with other ontologies, we can introduce additional semantics for augmenting an individual ontology's quality assurance [4, 6, 13].

Independent development of ontologies results in different naming schemes, leading to a challenge in alignment. For example, the class named "*lanugo*" in the SNOMED ontology is named as "*primary hair*" by the Foundational Model of Anatomy (FMA) ontology. Besides, real-world ontologies typically contain a large number of classes, which causes scalability issues during mapping discovery and makes it more difficult to distinguish classes of similar names (e.g., with overlapped sub-words) but distinct meanings, especially for systems that adopts string similarity-based lexical matching.

Leading systems such as LogMap [8] and AgreementMakerLight (AML) [5] approach alignment as a sequential process, with lexical matching typically being the first stage, followed by mapping extension and mapping repair. However, their lexical matching parts mostly take the text's surface form, without considering the semantics of words. More recent machine learning-based approaches such as DeepAlignment [10] and OntoEmma [15] adopt the word embedding

technique which projects words into vectors, where pairs of words with closer semantic meanings have a smaller Euclidean distance in the vector space. Nevertheless, these methods adopt traditional non-contextual word embedding methods which assign each word a unified representation and thus cannot well exploit word-level contexts to help resolve ambiguity.

To tackle this problem, we propose an ontology alignment system based on BERT, a contextual word embedding model that has demonstrated its strength (through fine-tuning) in a wide range of Natural Language Processing (NLP) tasks such as question answering, named entity recognition, and sentiment analysis [2, 16, 18], but has not yet been fully investigated in ontology alignment. The fundamental challenge of applying deep learning techniques on ontology alignment is that the number of reference mappings is often several orders smaller than the number of candidates (i.e., class pairs) to predict, resulting in a lack of labelled data and an imbalance between positive and negative samples. Thus, previous research into supervised learning schemes usually involves complicated feature engineering and needs to address extra noise brought by silver data (i.e. labelled data that are automatically generated by certain heuristics) [7, 15]. In contrast, fine-tuning the pretrained BERT on downstream tasks typically necessitates only a moderate amount of training data and avoids complex hand-crafted features. Furthermore, we also consider a critical issue in mapping prediction, i.e., reducing the quadratic complexity of searching all possible mappings.

To the best of our knowledge, we are among the first to develop a robust and general ontology alignment system using contextual embedding. In comparison to the previous work by Neutel et al. [12], which is a preliminary work that employs BERT to match two domain ontologies, we establish a concrete and flexible pipeline that fits both the unsupervised and semi-supervised settings; we improve their mean token embedding and the class token embedding models and use them as baselines; we optimize the mapping search, and conduct more extensive experiments to examine our approaches. We refer to our system as BERTMap[3]. As shown in Figure 1, it consists of the following steps:

1. **Corpora construction**. We extract *synonym* and *non-synonym* pairs from various sources, including input ontologies, known mappings and/or external knowledge. Such construction exploits the text semantics and avoids the need for hand-crafted features.
2. **Fine-tuning**. We then choose a suitable pretrained BERT variant, and fine-tune it on our corpora for a classifier, which takes a moderate amount of data and training resources.
3. **Mapping prediction**. For each class pair, we take their *labels* as input to the classifier. Since one class may have multiple labels, we use the average of the output probabilities of all the label combinations as the mapping value. To reduce the search space but keep the recall, we use a *sub-word inverted index* based on BERT's tokenizer.

[3] Code is available at: https://github.com/KRR-Oxford/BERTMap.

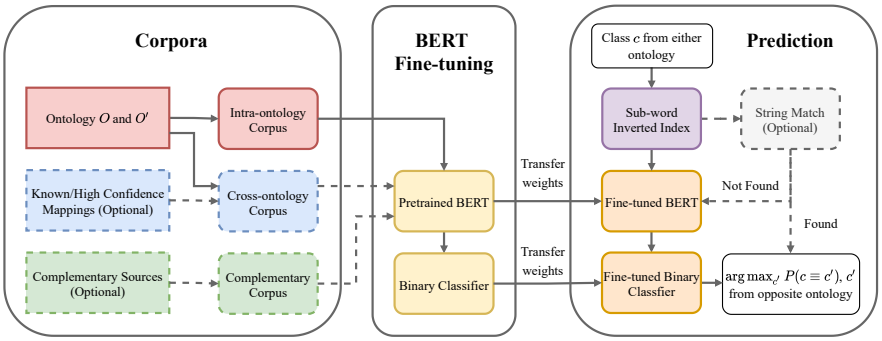

Fig. 1. Illustration of BERTMap system.

We evaluate BERTMap on *(i)* the FMA-SNOMED small fragment task of the OAEI Large BioMed Track (LargeBio)[4], and *(ii)* its extended version FMA-SNOMED+, where the missing labels of SNOMED are augmented with labels from a more recent version of SNOMED. We compare BERTMap with four internal baselines — two lexical matching-based methods and two BERT token embedding-based models, and two leading systems — LogMap [8] and AML [5]. Our results demonstrate that BERTMap, despite using the to-be-aligned classes alone as the input, outperforms all the baselines on both tasks.

2 Preliminaries

2.1 Problem Formulation

An ontology is typically defined as an explicit specification of a conceptualization. It often uses representational vocabularies to describe a domain of interest with the main components being *entities* and *axioms*. Note that entities include *classes*, *instances* and *properties*. Ontology alignment involves tasks of matching cross-ontology entities with *equivalence*, *subsumption* or other more complicated relationships. In this work, we focus on equivalence alignment between classes.

The ontology alignment system takes as input a pair of ontologies, O and O', with class sets C and C', respectively. It first generates a set of scored mappings, which are triples of the form of $(c \in C, c' \in C', P(c \equiv c'))$, where $P(c \equiv c') \in [0, 1]$ denotes the probability score (a.k.a. mapping value) that c and c' are equivalent. In this paper, we determine the final output by preserving mappings with a score larger than a certain threshold $\lambda \in [0, 1]$.

We further clarify some notations used in this paper. Note that a class of an ontology typically contains a list of labels (via annotation properties such as *rdfs:label*) that serve as alternative class names. We lowercase these aliases and remove any underscores before tokenization. We denote the preprocessed labels as ω and the set of them as $\Omega(c)$ for a class c.

[4] http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/.

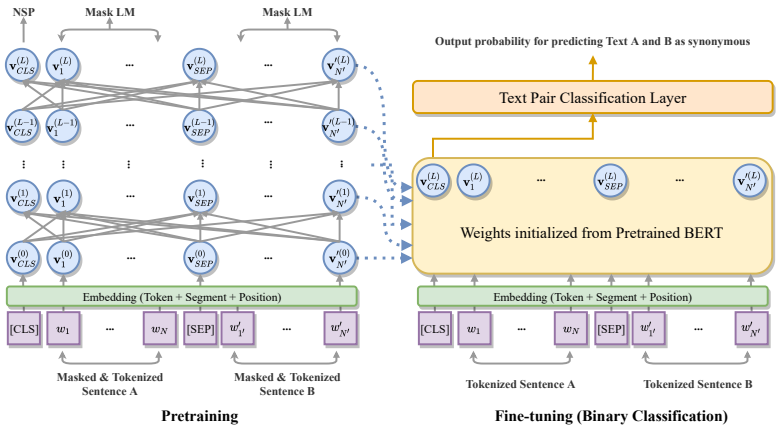

Fig. 2. BERT pretraining (left) and fine-tuning (right) for binary classification.

2.2 BERT: Pretraining and Fine-tuning

BERT is a language representation model built on the bidirectional transformer encoder [14]. As shown in Figure 2, its input is a sequence composed of a special token `[CLS]`, tokens of two sentences A and B, and a special token `[SEP]` that separates A and B. Each token's initial embedding encodes its content, its position in the sequence, and the sentence it belongs to. The model has L successive layers of an identical architecture. Its main component is the *multi-head self-attention* block, which computes a contextual hidden representation for each token by considering the *whole* sequence output from the previous layer (see the grey arrows in Figure 2). The output of layer l is denoted as:

$$f_{bert}(\mathbf{x}, l) = (\mathbf{v}_{CLS}^{(l)}, \mathbf{v}_1^{(l)}, ..., \mathbf{v}_N^{(l)}, \mathbf{v}_{SEP}^{(l)}, \mathbf{v}_1'^{(l)}, ..., \mathbf{v}_{N'}'^{(l)}) \in \mathbb{R}^{(N+N'+2)\times d} \quad (1)$$

where $\mathbf{x}$ is the input sequence, $\mathbf{v}_i^{(l)}$s and $\mathbf{v}_j'^{(l)}$s are d-dimensional vectors of corresponding tokens. The last layer ($l = L$) outputs can be used as the input of downstream tasks or as the token embeddings. In contrast to the traditional non-contextual word embedding techniques such as Word2Vec [11], which assign each token in the vocabulary only one embedding, BERT can distinguish different occurrences of the same token. For instance, given an input sentence *"the bank robber was seen on the river bank"*, BERT computes different embeddings for the two occurrences of *"bank"*, while the traditional model yields a unified embedding that is biased towards the most frequent meaning of *"bank"* (probably the money bank) in the training corpora.

The framework of BERT involves *pre-training* and *fine-tuning*, where pretraining is to develop a multi-purpose model that learns vast background knowledge, and fine-tuning is to adjust the parameters of pre-trained BERT by further training on a downstream task. On the left of Figure 2, we illustrate that BERT

is pre-trained on two tasks: Masked Language Modelling (MLM) which predicts tokens that are randomly *masked* in sentences A and B, and Next Sentence Prediction (NSP) which predicts if sentence B follows A. On the right of Figure 2, we present the example of fine-tuning on a downstream paraphrasing task, where the pre-trained BERT is attached to an additional binary classification layer that outputs the probability that A and B are synonymous. By minimizing the cross-entropy loss on the training samples, the parameters of pre-trained BERT are adjusted, and the parameters of the additional layer are learnt. Pre-trained BERT models are usually publicly available and can be re-used for fine-tuning on various downstream tasks. In this paper, we conduct no pre-training but instead fine-tuning an existing pre-trained BERT that has learnt biomedical background knowledge (see Section 4 for details).

3 BERTMap

3.1 Corpora Construction and Fine-tuning

Real-world ontologies are typically abundant in labels that serve as aliases to their classes. Labels of the same class or semantically equivalent classes are intuitively synonymous in the domain of the input ontologies. On the other hand, non-synonymous pairs can be extracted from classes that are semantically distinct. For convenience, we use "*synonym*" and "*non-synonym*" to describe a synonymous and a non-synonymous label pair, respectively. The corpora of these pairs are divided into the following three categories:

Intra-ontology corpus. For each input ontology, we regard each pair of labels associated with the same class as synonymous. We also construct *identity synonyms* to encode each label as a synonym of itself. For non-synonyms, we consider: *soft non-synonyms* which are labels from separate classes at random, and *hard non-synonyms* which are labels from *disjoint* classes. Since class disjointness is rarely defined in an ontology, we infer it from the structure of the input ontology. In this paper, we simply assume that *sibling* classes are disjoint.

Cross-ontology corpus. The lack of annotated mappings makes it unfeasible to apply supervised learning on ontology alignment. However, we can optionally employ a semi-supervised setting by assuming that a small portion of mappings have been created by human experts. For each known mapping, we extract label pairs from its two classes as synonyms. Meanwhile, soft non-synonyms are extracted by matching a source class to a random target class, whereas hard non-synonyms are not available at the cross-ontology level because we have no predefined disjointness in the mappings. Also, we do not create identity synonyms here because they have been considered in the intra-ontology corpus.

Complementary corpus. Besides the input ontologies, we can expand the synonym and non-synonym sets from external sources, especially other ontologies in the relevant domain. To reduce the potential noise and the corpus size, we could truncate the auxiliary ontology by considering only the classes whose labels can be matched to some class of the input ontologies.

The intra-ontology corpus, cross-ontology corpus and complementary corpus are denoted as *io*, *co* and *cp*, respectively. *io* is essential to BERTMap, while *co* and *cp* are optional. The identity synonyms are denoted as *ids*. For convenience, we use + to denote the combination of different corpus/synonyms; for example, $io + ids$ refers to the intra-ontology corpus with identity synonyms considered, and *io*+*co*+*cp* refers to including all three corpora without identity synonyms. To learn the symmetrical property, we also consider appending reversed synonyms, i.e., if (ω_1, ω_2) is in the synonym set, (ω_2, ω_1) is also added as a synonym. Given a corpus setting, we obtain the corresponding synonym and non-synonym sets, and then fine-tune a pretrained BERT on them as introduced in Section 2.2. We evaluate various corpus settings in Section 4. Finally, since some non-synonym pairs are extracted by random class combination, they can occasionally appear in the synonym set; in such cases we delete the relevant non-synonym pair.

3.2 Candidate Selection with Sub-word Inverted Index

Given input ontologies O and O', and their class sets C and C', a naive algorithm for computing the alignment is to look up $c' = \arg\max_{c' \in C'} P(c \equiv c')$ for every class $c \in C$, which results in a time complexity of $O(n^2)$. To reduce that, we use a *sub-word inverted index* based on BERT's WordPiece tokenizer [17]. The algorithm first initializes the vocabulary with single characters present in the training corpus and incrementally merges them into sub-words so that the most likely combination is added at each iteration. We opt to use the built-in sub-word tokenizer rather than re-train it on our corpora because it has already been fitted to an enormous corpus (with 3.3 billion words) that covers various topics [3], and in this context we consider generality to be preferable to task specificity.

We build sub-word inverted indices for O and O' separately. Each entry of an index is a sub-word, and its values are classes whose labels contain this sub-word after tokenization. With the indices, we can implement candidate selection very efficiently in the following way. We first restrict the search space of each source class c to target classes that share at least one sub-word token with c. Next, we rank these target classes by a scoring metric based on inverted document frequency (*idf*), and the top k scored are chosen for subsequent mapping prediction. For a target class c', this scoring metric is computed as:

$$s(c, c') = \sum_{t \in T(c) \cap T(c')} idf(t) = \sum_{t \in T(c) \cap T(c')} \log_{10}(|C'| \,/\, |C'(t)|),$$

where $T(\cdot)$ is the set of sub-word tokens from tokenzing all the labels of a class, $C'(t)$ is the the set of target classes that have token t after tokenization, and $|\cdot|$ denotes set cardinality. In this way, we reduce the search space from $O(n^2)$ to $O(kn)$ where k is a constant. Compared to the traditional word-level inverted index, our approach has the following advantages: *(i)* it captures various forms of words without requiring additional processing such as stemming and consulting a dictionary; *(ii)* it interprets unknown words by parsing them into consecutive known sub-words rather than treating them as the same (unknown) token.

3.3 Mapping Prediction

With the ranked candidate classes of a source class c, we first perform a *string-match* check to see whether any of the candidate classes have at least one exactly matched label (after preprocessing as illustrated in Section 2.1) with c, i.e., to search for c' according to the rank such that $\Omega(c) \bigcap \Omega(c') \neq \emptyset$. We assign a mapping value of 1.0 to the first c' that satisfies the condition. If we cannot find such candidate class, we apply the fine-tuned BERT classifier. In this case, for each candidate class c', we predict the synonym probabilities for all label pairs $(\omega, \omega') \in \Omega(c) \times \Omega(c')$, and take the average as the mapping value between c and c'. We return the top scored mapping for each c.

We can generate three mapping sets from the input source and target ontologies: *(i)* `src2tgt` by looking for a target class $c' \in C'$ for each source class $c \in C$; *(ii)* `tgt2src` by looking for a source class $c \in C$ for each target class $c' \in C'$; and *(iii)* `combined` by merging `src2tgt` and `tgt2src` with duplicates removed. We finally output determined mappings by filtering out mappings whose values are lower than a certain threshold $\lambda \in [0, 1]$.

4 Experiments

4.1 Datasets and Experiment Settings

Datasets and Tasks. We first evaluate BERTMap on the FMA-SNOMED small fragment task of the OAEI LargeBio Track. The input FMA and SNOMED ontologies (segments) have $10,157$ and $13,412$ classes, respectively. The dataset also includes a set of UMLS-based reference (ground truth) mappings for evaluating the systems, with $6,026$ of them marked by "=" and $2,982$ marked by "?". Mappings marked by "?" will cause logical conflicts after alignment, so they are regarded as neither positive nor negative in evaluation. We construct the complementary corpus for FMA-SNOMED task by utilizing class labels from the most recent version of the original SNOMED[5]. Note that the complementary labels are functional in fine-tuning but not prediction. To examine the scenario when baseline systems can also use these additional labels, we consider the extended task FMA-SNOMED+, where the input ontology SNOMED is extended to SNOMED+ by incorporating these labels.

BERTMap Settings. We set up different corpus settings for training. Recall the corpus notations in Section 3.1, the unsupervised and semi-supervised learning settings are distinguished by including *co* (cross-ontology corpus) or not, and *io* (intra-ontology corpus) is always considered. In the unsupervised learning setting, 80% of the fine-tuning corpus are used for training and 20% for validation. The final mapping prediction is evaluated on the full set of reference mappings. In the semi-supervised learning setting, the training data is formed by incorporating all the unsupervised fine-tuning data and *co* constructed from 20% of the reference mappings. We use an additional *co* constructed from 10% of the

[5] The version of 20210131 from https://www.nlm.nih.gov/healthit/snomedct/index.html.

reference mappings as the validation set. We take the remaining 70% as the test mappings for evaluating mapping prediction. Note that here validation is different from testing because the former concerns fine-tuning while the latter concerns mapping prediction. We also examine the impact of *ids* (identity synonyms) on both tasks, and the impact of *cp* (complementary corpus) on FMA-SNOMED. In implementation, we consider all the synonyms in the positive sample set, and randomly sample 2 soft non-synonyms and 2 hard non-synonyms for each synonym in *io*, and 4 soft non-synonyms for each synonym in *co*. We perform the same negative sampling procedure on *cp* as on *io* because *cp* is also a corpus derived from one (external) ontology. As a result, the positive-negative ratio is consistently $1:4$ for all the corpus settings. For settings that consider *ids*, we sample the corresponding number of non-synonyms to keep this ratio.

We adopt Bio-Clinical BERT which has been pretrained on biomedical and clinical domain corpora [1]. We fine-tune the BERT model for 3 epochs with a batch size of 32, and evaluate it on the validation set for every 0.1 epoch, through which the best checkpoint (on the cross-entropy loss) is selected for testing. The input maximum length is set to 128. In prediction, the number of candidates selected using the sub-word inverted index is set to 200. Our implementation uses `owlready2`[6] and `transformers`[7].

Baselines. We compare BERTMap with the following baselines:

1. *String-match*. It sets the mapping value of two classes c and c' to 1.0 if $\Omega(c) \bigcap \Omega(c') \neq \emptyset$, and to 0 otherwise.
2. *Edit-similarity*. Given a source class c, it predicts the target class c' and the corresponding mapping value by $\arg\max_{c' \in \zeta(c)} \mathbf{nes}(\Omega(c), \Omega(c'))$, where $\mathbf{nes}(\cdot, \cdot)$ refers to the *maximum* normalized edit similarity between the labels of c and c', and $\zeta(c)$ denotes the candidates of c that are selected in the same way as BERTMap. Note that *string-match* is a special case of *Edit-similarity*.
3. *Mean-embeds* and *Cls-embeds*. BERT outputs token embeddings $f_{bert}(\mathbf{x}, l)$ at layer l (see (1)). Mean-embeds (the mean token embedding model) extracts the mean of all the token embeddings of the last layer L, denoted as $\overline{f_{bert}(\mathbf{x}, L)}$, as the embedding of a class label, and calculates the *cosine similarity* of two classes as their mapping value. Note that the embeddings of multiple labels of a class are averaged. Cls-embeds (the class token embedding model) is the same except that it considers the class token embedding of the last layer, i.e., $\mathbf{v}^{(L)}_{CLS}$, as a class label's embedding. As in BERTMap, *string-match* is also first considered before calculating the cosine similarity.
4. *LogMap*, *AML* and *LogMapLt*. LogMap and AML are two lexical matching and reasoning based systems with leading performance in many OAEI tracks and other tasks. Since LogMap and AML consider the neighbourhoods and relevant logical axioms of two classes while BERTMap at the current stage only considers the class labels, we additionally introduce LogMapLt, which only uses the lexical matching part of LogMap, for comparison.

[6] https://owlready2.readthedocs.io/en/latest/.
[7] https://huggingface.co/transformers/.

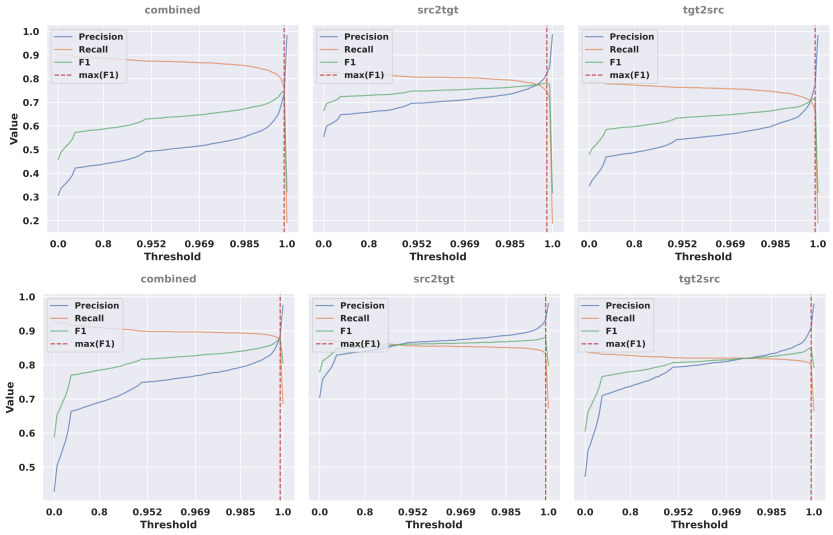

Fig. 3. Precision, Recall and Macro-F1 of BERTMap as the mapping value threshold λ ranges from 0 to 1. The top three figures are for the *io+co+ids* setting on FMA-SNOMED task, and the bottom three are for the *io+ids* setting on FMA-SNOMED+ task. The maximum F1 is indicated by a red vertical line.

4.2 Results

We illustrate the resulting Precision, Recall and Macro-F1 scores in Table 1 and 2. Note that, in the semi-supervised learning setting, we measure the performance only on the test mappings. We search for the optimal combination of mapping set (`src2tgt`, `tgt2src` and `combined`) and the corresponding mapping value threshold λ that leads to the best Macro-F1 score on the validation mappings (10%) for the semi-supervised models. We select the best combination on full mappings for the unsupervised models due to the shortage of validation mappings. Nevertheless, we will later illustrate that BERTMap models are robust to mapping threshold selection, and we will obtain similar results if a reasonable validation set is provided. For the baselines, we also select the best mapping set-threshold combination for each of them.

The overall results show that BERTMap attains the best F1 score (typically with high recall) among all the systems for both tasks. On the FMA-SNOMED task, the best unsupervised BERTMap model surpasses AML (resp. LogMap) by 2.0% (resp. 4.6%) in F1, while the best semi-supervised BERTMap model exceeds AML (resp. LogMap) by 3.7% (resp. 6.1%). The corresponding statistics become 1.8% (resp. 1.0%) and 2.9% (resp. 2.3%) on the FMA-SNOMED+ task.

The string-match and edit-similarity baselines perform much better on the FMA-SNOMED+ task than the FMA-SNOMED task because they rely on the sufficiency of class labels in input ontologies, whereas BERTMap can learn from

		Full Mappings			Test Mappings		
	System	Precision	Recall	Macro-F1	Precision	Recall	Macro-F1
Unsupervised	io	0.321	0.625	0.424	0.248	0.621	0.354
	io+ids	0.635	0.727	0.678	0.561	0.704	0.625
	io+cp	0.862	0.822	**0.842**	0.867	0.786	0.825
	io+cp+ids	0.860	0.824	**0.842**	0.866	0.782	0.822
Semi-supervised	io+co	NA	NA	NA	0.822	0.773	0.797
	io+co+ids	NA	NA	NA	0.821	0.747	0.782
	io+co+cp	NA	NA	NA	0.839	0.824	0.832
	io+co+cp+ids	NA	NA	NA	0.875	0.813	**0.843**
Baselines	string-match	0.988	0.196	0.328	0.983	0.192	0.321
	edit-similarity	0.523	0.386	0.444	0.430	0.378	0.402
	mean-embeds	0.464	0.500	0.481	0.422	0.450	0.436
	cls-embeds	0.522	0.242	0.331	0.970	0.192	0.321
	AML	0.902	0.758	0.824	0.865	0.754	0.806
	LogMap	0.942	0.689	0.796	0.918	0.681	0.782
	LogMapLt	0.969	0.208	0.342	0.956	0.204	0.336

Table 1. BERTMap and baseline results on the FMA-SNOMED task.

		Full Mappings			Test Mappings		
	System	Precision	Recall	Macro-F1	Precision	Recall	Macro-F1
Unsupervised	io	0.893	0.874	**0.883**	0.911	0.834	0.871
	io+ids	0.932	0.833	0.880	0.906	0.832	0.868
Semi-supervised	io+co	NA	NA	NA	0.913	0.841	**0.875**
	io+co+ids	NA	NA	NA	0.913	0.836	0.873
Baselines	string-match	0.975	0.686	0.805	0.964	0.678	0.796
	edit-similarity	0.965	0.750	0.844	0.950	0.746	0.836
	mean-embeds	0.972	0.690	0.807	0.960	0.683	0.798
	cls-embeds	0.972	0.686	0.805	0.963	0.678	0.796
	AML	0.905	0.828	0.865	0.868	0.825	0.846
	LogMap	0.880	0.865	0.873	0.838	0.868	0.852
	LogMapLt	0.958	0.718	0.821	0.940	0.709	0.808

Table 2. BERTMap and baseline results on the FMA-SNOMED+ task.

external resources. Edit-similarity is consistently better than string-match because it has already considered all the string-match cases, but it is still worse than LogMap and AML. Note that we also apply string-match for the mean-embeds and cls-embeds models before calculating the cosine similarity between the source and target classes' embeddings. Still, the results are merely better than the string-match baseline. This suggests that directly using pretrained BERT to encode class embeddings and calculate their distance in vector space is not adequate—we need fine-tuning to utilize the BERT embeddings effectively.

Compared to LogMap's lexical matcher, LogMapLt, BERTMap performs better than 50% on FMA-SNOMED and 6% on FMA-SNOMED+, implying the potential of BERTMap to become more powerful when it is extended to incorporate structural and logical information. For example, we can adjust the mapping

values by taking the alignment of neighbouring classes into account. We can also apply the reasoning-based ontology repair module [9] to prune the mapping set.

Regarding the BERTMap settings, we observe that when the input ontologies have sufficient labels (i.e., on the FMA-SNOMED+ task), considering intra-ontology corpus alone has already yielded promising results. Also, BERTMap has better performance when it incorporates the cross-ontology and complementary corpora—especially on the FMA-SNOMED task—where the SNOMED ontology is deficient in labels. Including the identity synonyms can also improve the performance, but not that prominent compared with without.

Finally, in Figure 3, we illustrate the effect of the mapping value threshold λ on BERTMap under two settings, i.e., $io + co + ids$ on FMA-SNOMED and $io + ids$ on FMA-SNOMED+. We can observe that in all cases the highest F1 scores are achieved when λ is very close to 1.0, and as λ increases, Precision grows significantly while Recall does not drop much. This suggests that BERTMap is robust to selecting an appropriate λ.

5 Conclusion and Discussion

In this study, we investigate ontology alignment using a contextual embedding-based model, BERTMap, that exploits the ontologies' text semantics. Rather than using a complex combination of machine learning and hand-crafted features, we construct a straightforward BERT fine-tuning task that learns the meanings of class labels, and we apply the resulting classifier to mapping prediction, which leads to promising results on two biomedical ontology alignment tasks. BERTMap is suitable for real-world applications because it supports both unsupervised and semi-supervised modes, and can well incorporate external materials when the input ontologies are incomplete in class labels. As part of our future work, we aim to develop the mapping extension and repair modules so as to make BERTMap a full-fledged ontology alignment system.

Acknowledgments

This work was supported by the SIRIUS Centre for Scalable Data Access (Research Council of Norway, project 237889), Samsung Research UK, Siemens AG, and the EPSRC projects AnaLOG (EP/P025943/1), OASIS (EP/S032347/1), UK FIRES (EP/S019111/1) and the AIDA project (Alan Turing Institute).

References

1. Alsentzer, E., Murphy, J., Boag, W., Weng, W.H., Jindi, D., Naumann, T., McDermott, M.: Publicly available clinical BERT embeddings. In: Proceedings of the 2nd Clinical Natural Language Processing Workshop. pp. 72–78 (Jun 2019)
2. Clark, C., Lee, K., Chang, M.W., Kwiatkowski, T., Collins, M., Toutanova, K.: BoolQ: Exploring the surprising difficulty of natural yes/no questions. In: Proceedings of NAACL-HLT. pp. 2924–2936 (2019)

3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT. pp. 4171–4186 (2019)
4. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Santos, C.: Ontology alignment evaluation initiative: Six years of experience. J. Data Semant. **15**, 158–192 (2011)
5. Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I.F., Couto, F.M.: The agreementmakerlight ontology matching system. In: Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., De Leenheer, P., Dou, D. (eds.) On the Move to Meaningful Internet Systems: OTM 2013 Conferences. pp. 527–541. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
6. Horrocks, I., Chen, J., Lee, J.: Tool support for ontology design and quality assurance (2020)
7. Iyer, V., Agarwal, A., Kumar, H.: Veealign: a supervised deep learning approach to ontology alignment. In: OM@ISWC (2020)
8. Jiménez-Ruiz, E., Cuenca Grau, B.: Logmap: Logic-based and scalable ontology matching. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) The Semantic Web – ISWC 2011. pp. 273–288. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
9. Jiménez-Ruiz, E., Meilicke, C., Grau, B.C., Horrocks, I.: Evaluating mapping repair systems with large biomedical ontologies. In: Description Logics (2013)
10. Kolyvakis, P., Kalousis, A., Kiritsis, D.: DeepAlignment: Unsupervised ontology matching with refined word vectors. In: Proceedings of NAACL-HLT. pp. 787–798 (Jun 2018)
11. Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR (2013)
12. Neutel, S., Boer, M.D.: Towards automatic ontology alignment using bert. In: AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering (2021)
13. Shvaiko, P., Euzenat, J.: Ontology matching: State of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering **25**(1), 158–176 (2013)
14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)
15. Wang, L., Bhagavatula, C., Neumann, M., Lo, K., Wilhelm, C., Ammar, W.: Ontology alignment in the biomedical domain using entity definitions and context. In: Proceedings of the BioNLP 2018 workshop. pp. 47–55 (Jul 2018)
16. Wu, Q., Lin, Z., Karlsson, B., Lou, J.G., Huang, B.: Single-/multi-source cross-lingual NER via teacher-student learning on unlabeled data in target language. In: Proceedings of ACL. pp. 6505–6514 (Jul 2020)
17. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., Dean, J.: Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR (2016)
18. Yin, D., Meng, T., Chang, K.W.: SentiBERT: A transferable transformer-based architecture for compositional sentiment semantics. In: Proceedings of ACL. pp. 3695–3706 (Jul 2020)