BERTMap: A BERT-based Ontology Alignment System

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

[1] Department of Computer Science, University of Oxford, UK
[2] Samsung Research, UK
{yuan.he,jiaoyan.chen,ian.horrocks}@cs.ox.ac.uk, denvar.a@samsung.com

Abstract

Ontology alignment (a.k.a ontology matching (OM)) plays a critical role in knowledge integration. Owing to the success of machine learning in many domains, it has been applied in OM. However, the existing methods, which often adopt ad-hoc feature engineering or non-contextual word embeddings, have not yet outperformed rule-based systems especially in an unsupervised setting. In this paper, we propose a novel OM system named BERTMap which can support both unsupervised and semi-supervised settings. It first predicts mappings using a classifier based on fine-tuning the contextual embedding model BERT on text semantics corpora extracted from ontologies, and then refines the mappings through extension and repair by utilizing the ontology structure and logic. Our evaluation with three alignment tasks on biomedical ontologies demonstrates that BERTMap can often perform better than the leading OM systems LogMap and AML.

Introduction

Ontology alignment (a.k.a. ontology matching (OM)) aims at matching semantically related entities from different ontologies. A relationship (usually equivalence or subsumption) between two matched entities is known as a mapping. OM plays an important role in knowledge engineering, as a key technique for ontology integration and quality assurance (Horrocks, Chen, and Lee 2020; Shvaiko and Euzenat 2013). The independent development of ontologies often results in heterogeneous knowledge representations with different categorizations and naming schemes. For example, the class named *"muscle layer"* in the SNOMED Clinical Terms ontology is named *"muscularis propria"* in the Foundational Model of Anatomy (FMA) ontology. Moreover, real-world ontologies often contain a large number of classes, which not only causes scalability issues, but also makes it harder to distinguish classes with similar names and/or contexts but representing different objects.

Traditional OM solutions typically use lexical matching as their basis and combine it with structural matching and logic-based mapping repair. This has led to several classic systems such as LogMap (Jiménez-Ruiz and Cuenca Grau 2011) and AgreementMakerLight (AML) (Faria et al. 2013) which still demonstrate state-of-the-art performance on

alignment tasks. However, their lexical matching part only considers texts' surface form such as overlapped sub-strings, and cannot capture the word semantics. Recently, machine learning has been proposed as a replacement for lexical and structural matching; for example, DeepAlignment (Kolyvakis, Kalousis, and Kiritsis 2018) and OntoEmma (Wang et al. 2018) utilize word embeddings to represent classes and compute two classes' similarity according to their word vectors' Euclidean distance. Nevertheless, these methods adopt either traditional non-contextual word embedding models such as Word2Vec (Mikolov et al. 2013), which only learns a global (context-free) embedding for each word, or use complex feature engineering which is ad-hoc and relies on a large number of annotated samples for training. In contrast, pre-trained transformer-based language representation models such as BERT (Devlin et al. 2019) can learn robust contextual text embeddings, and usually require only moderate training resources for fine-tuning. Although these models perform well in many Natural Language Processing tasks, they have not yet been sufficiently investigated in OM.

In this paper, we propose BERTMap, a novel ontology alignment system that exploits BERT fine-tuning for mapping prediction and utilizes the graphical and logical information of ontologies for mapping refinement. As shown in Figure 1, BERTMap includes the following main steps: *(i) corpus construction*, where synonym and non-synonym pairs from various sources are extracted; *(ii) fine-tuning*, where a suitable pre-trained BERT model is selected and fine-tuned on the corpora constructed in *(i)*; *(iii) mapping prediction*, where mapping candidates are first extracted based on sub-word inverted indices and then predicted by the fine-tuned BERT classifier; and *(iv) mapping refinement*, where additional mappings are recalled from neighbouring classes of highly scored mappings, and some mappings that lead to logical inconsistency are deleted for higher precision.

We evaluate BERTMap[1] on the FMA-SNOMED task and the FMA-NCI task of the OAEI Large BioMed Track[2], and an extended task of FMA-SNOMED where the more complete labels from the original SNOMED ontology are added. Our results demonstrate that BERTMap can often outperform the state-of-the-art systems LogMap and AML.

---

[1] Codes and data: https://github.com/KRR-Oxford/BERTMap.
[2] http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/

Preliminaries

Problem Formulation

An ontology is mainly composed of entities (including classes, instances and properties), and axioms that can express relationships between entities. Ontology alignment involves identifying equivalence, subsumption or other more complex relationships between cross-ontology pairs of entities. In this work, we focus on equivalence between classes. Given a pair of ontologies, O and O', whose named class sets are C and C', respectively, we aim to first generate a set of scored mappings of the form $(c \in C, c' \in C', P(c \equiv c'))$, where $P(c \equiv c') \in [0, 1]$ is a score indicating the degree to which c and c' are equivalent; we then extend and repair the scored mappings to output determined mappings.

BERT: Pre-training and Fine-tuning

BERT is a contextual language representation model built on bidirectional transformer encoders (Vaswani et al. 2017). Its framework involves *pre-training* and *fine-tuning*. In pre-training, the input is a sequence composed of a special token [CLS], tokens of one sentence A, a special token [SEP], and tokens of another sentence B that follows A in the corpus. Each token's initial embedding encodes its content, its position in the sequence, and the sentence it belongs to (A or B). The model has multiple successive layers of an identical architecture. Its main component is the multi-head self-attention block which computes a contextual hidden representation of each token by considering the output of the whole sequence from the previous layer. The tokens' embeddings from the last layer can be used as the input of a downstream task. Pre-training is conducted by minimizing losses on two tasks: Masked Language Modelling, which predicts a part of tokens that are randomly *masked*, and Next Sentence Prediction, which predicts whether sentence B follows A. In contrast to the traditional non-contextual word embedding methods which assign each token only one embedding, BERT distinguishes different occurrences of the same token. For instance, given a sentence *"the bank robber was seen on the river bank"*, BERT computes different embeddings for the two occurrences of *"bank"*, while a non-contextual model yields a unified embedding that is biased towards the most frequent meaning in the corpus. In fine-tuning, pre-trained BERT is attached to customized downstream layers and takes as input either one sentence (e.g., for sentiment classification) or two sentences (e.g., for paraphrasing) according to specific tasks. It typically necessitates only a few epochs and a moderate number of samples for training.

BERTMap

The workflow of BERTMap is shown in Figure 1. In this section, we describe the technical details of each step.

Corpus Construction and BERT Fine-tuning

Text Semantics Corpora In real-world ontologies, a named class often has multiple labels (aliases) defined by annotation properties such as *rdfs:label*. For convenience, we denote a label after preprocessing[3] by ω, and denote the set of all the preprocessed labels of a class c as $\Omega(c)$. Labels of the same class or from semantically equivalent classes are intuitively synonymous in the domain of the input ontologies; labels from semantically distinct classes can be regarded as non-synonymous. The corpora for BERT fine-tuning are composed of pairs of such synonymous labels (i.e., "*synonyms*") and pairs of such non-synonymous labels (i.e.,"*non-synonyms*"). According to the source, the corpora are divided into three categories as follows.

Intra-ontology corpus. For each named class c in an input ontology we derive all its synonyms, which are pairs (ω_1, ω_2) with $\omega_1, \omega_2 \in \Omega(c)$, and the special cases where $\omega_1 = \omega_2$ are referred to as *identity synonyms*. We consider two types of non-synonyms: *(i) soft non-synonyms* which are labels from two random classes; and *(ii) hard non-synonyms* which are labels from logically disjoint classes. Since class disjointness is often not defined in an ontology, we simply assume that sibling classes (i.e., classes that share a common superclass) are disjoint. In fact, this is a naive solution to infer disjointness from the structure of the input ontology.

Cross-ontology corpus. The lack of annotated mappings makes it unfeasible to apply supervised learning on ontology alignment. However, it is reasonable to support a semi-supervised setting where a small portion of annotated mappings are given and we can extract synonyms from these mappings. Given a mapping composed of two named classes c and c' we extract all synonyms (ω, ω') where $(\omega, \omega') \in \Omega(c) \times \Omega(c')$ ($\times$ refers to the Cartesian Product). We also extract non-synonyms from pairs of randomly aligned classes.

Complementary corpus. We can optionally utilize auxiliary ontologies for additional synonyms and non-synonyms. They are extracted in the same way as the intra-ontology corpus but from an auxiliary ontology. To reduce data noise and limit the corpus size, we consider auxiliary ontologies of the same domain and only utilize named classes that have shared labels with some class of the input ontologies.

The intra-ontology, cross-ontology and complementary corpora are denoted as io, co and cp, respectively. 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 append reversed synonyms, i.e., if (ω_1, ω_2) is in the synonym set, (ω_2, ω_1) is added. Since some non-synonyms are extracted randomly, they can occasionally also appear in the synonym set; in this case, we delete the non-synonyms.

Fine-tuning Given sets of synonyms and non-synonyms as positive and negative samples, respectively, we fine-tune a pre-trained BERT along with a downstream binary classifier on the cross-entropy loss. Note we conduct no pre-training but use an existing one from the Hugging Face library[4]. The inputs of BERT are the tokenized label pairs with the

---

[3]This includes lowercasing and underscore symbol removing.
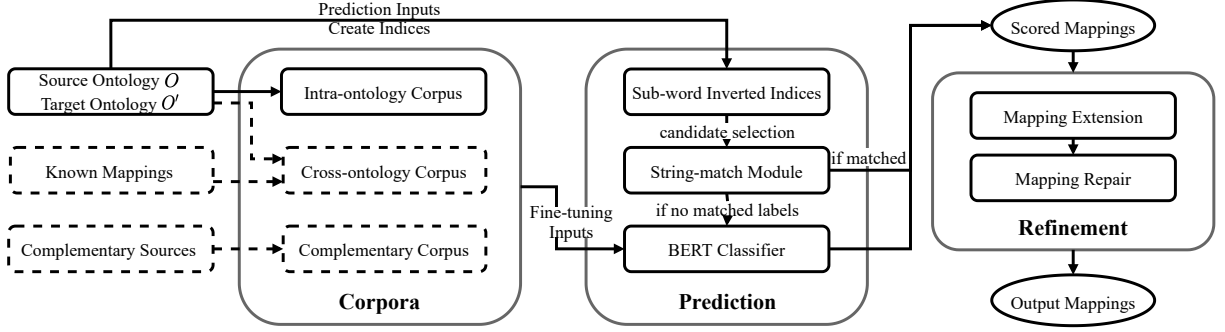[4]https://huggingface.co/models

Figure 1: Illustration of BERTMap system.

maximum length set to 128. The classifier consists of a linear layer (with dropout) that takes as input the embedding of `[cls]` token from BERT's last-layer outputs, and transforms it into a 2-dimensional vector before applying the output *softmax* layer. The optimization is done using the Adam algorithm (Loshchilov and Hutter 2017). The final output is of the form $\langle 1 - s, s \rangle$, where $s \in [0, 1]$ is the score that indicates the degree that the input label pairs are synonymous.

Mapping Prediction

To compute a matched class for each class $c \in C$, a naive solution is to search for $\arg\max_{c' \in C'} P(c \equiv c')$. Computing mappings in this way has a time complexity of $O(n^2)$, which is impractical for matching large ontologies. To reduce the search space, BERTMap first selects a set of candidates of matched classes using sub-word inverted indices, and then scores each potential mapping with the fine-tuned BERT.

Candidate Selection The assumption of our candidate selection is that matched classes are likely to have labels with overlapped sub-tokens. Previous works typically adopt word-level inverted index with additional text processing such as stemming and dictionary consulting (Jiménez-Ruiz and Cuenca Grau 2011; Wang et al. 2018). In contrast, BERTMap exploits sub-word inverted index which can *(i)* capture various word forms without extra processing, and *(ii)* parse unknown words into consecutive known sub-words instead of simply treating them as one special token.

We build sub-word inverted indices based on BERT's inherent WordPiece tokenizer (Wu et al. 2016), which is trained by an incremental procedure that merges characters (from the corpus) into most likely sub-words 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 (Devlin et al. 2019), and in this context we consider generality to be preferable to task specificity.

We construct[5] indices I and I' for the two input ontologies O and O', respectively. Each entry of an index is a sub-word, and its values are classes that have at least one label containing this sub-word after tokenization. A query of source (resp. target) classes that contain a token t is denoted

---

[5]Index construction is linear w.r.t. the number of sub-words.

as $I[t]$ (resp. $I'[t]$). The function that takes a class as input and returns all the sub-word tokens of this class's labels is denoted as $T(\cdot)$. Given a source class c, we search from C' the target candidate classes as follows: we first select target classes that share at least one sub-word token with c, i.e., $\bigcup_{t \in T(c)} I'[t]$, and then rank them according to a scoring metric based on *inverted document frequency (idf)*:

$$S_{select}(c, c') = \sum_{t \in T(c)} idf(t) = \sum_{t \in T(c)} \log_{10} \frac{|C'|}{|I'[t]|}$$

where $| \cdot |$ denotes set cardinality. Finally, we choose the top k scored target classes for c to form potential mappings of which the scores will be computed. As a result, we reduce the quadratic time complexity to $O(kn)$ where $k << n$ is the cut-off of candidate selection.

Mapping Score Computation For a matched class candidate c' of the source class c, BERTMap uses string matching and the fine-tuned BERT classifier to calculate the mapping score between them as follows:

$$S_{map}(c, c') = \begin{cases} 1.0 & \text{if } \Omega(c) \bigcap \Omega(c') \neq \emptyset \\ S_{bert}(\Omega(c), \Omega(c')) & \text{otherwise} \end{cases}$$

where $\Omega(c) \bigcap \Omega(c') \neq \emptyset$ means c and c' have at least one exactly matched label. $S_{bert}(\cdot, \cdot)$ denotes the average of the synonym scores of all the label pairs (i.e., $(\omega, \omega') \in \Omega(c) \times \Omega(c')$), which are predicted by the BERT classifier. The purpose of the string-matching is to save computation by avoiding unnecessary use of the BERT classifier on "easy" mappings. BERTMap finally returns the mapping for c by selecting the top scored candidate $c' = \arg\max S_{map}(c, c')$.

With the above steps, we can optionally generate three sets of scored mappings: *(i)* `src2tgt` by looking for a matched target class $c' \in C'$ for each source class $c \in C$; *(ii)* `tgt2src` by looking for a matched source class $c \in C$ for each target class $c' \in C'$; and *(iii)* `combined` by merging `src2tgt` and `tgt2src` with duplicates removed. We denote the hyperparameters as τ and λ where τ refers to the set type (`src2tgt`, `tgt2src` or `combined`) of scored mappings and $\lambda \in [0, 1]$ refers to the mapping score threshold.

Mapping Refinement

Mapping Extension If a source class c and a target class c' are matched, their respective semantically related classes

Algorithm 1: Iterative Mapping Extension

Input: High confidence mapping set, $\mathcal{M}$
Parameter: Extension threshold, κ
Output: Extended mapping set, $\mathcal{M}_{\text{ex}}$

1: Initialize the frontier: $\mathcal{M}_{fr} \leftarrow \mathcal{M}$
2: Initialize the extended mapping set: $\mathcal{M}_{ex} \leftarrow \{\}$
3: Let $\text{Sup}(\cdot)$ be the function that returns superclasses
4: Let $\text{Sub}(\cdot)$ be the function that returns subclasses
5: **while** $\mathcal{M}_{fr}$ is not empty **do**
6:    Initialize an empty new extension set: $\mathcal{M}_{new} \leftarrow \{\}$
7:    **for** each mapping $(c, c', S_{map}(c, c')) \in \mathcal{M}_{fr}$ **do**
8:       **for** $(x, x') \in (\text{Sup}(c) \times \text{Sup}(c')) \cup (\text{Sub}(c) \times \text{Sub}(c'))$ **do**
9:          $m \leftarrow (x, x', S_{map}(x, x'))$
10:          **if** $S_{map}(x, x') \geq \kappa$ **and** $m \notin \mathcal{M}$ **and** $m \notin \mathcal{M}_{ex}$ **then**
11:             $\mathcal{M}_{new} \leftarrow \mathcal{M}_{new} \cup \{m\}$
12:          **end if**
13:       **end for**
14:    **end for**
15:    $\mathcal{M}_{ex} \leftarrow \mathcal{M}_{ex} \cup \mathcal{M}_{new}$
16:    $\mathcal{M}_{fr} \leftarrow \mathcal{M}_{new}$
17: **end while**
18: **return** $\mathcal{M}_{ex}$

Task	SRC	TGT	Refs (=)	Refs (?)
FMA-SNOMED	10,157	13,412	6,026	2,982
FMA-NCI	3,696	6,488	2,686	338

Table 1: Numbers of classes and reference mappings in the FMA-SNOMED and FMA-NCI tasks.

gorithm (Jiménez-Ruiz et al. 2013).

Evaluation

Experiment Settings

Datasets and Tasks The evaluation considers the FMA-SNOMED and FMA-NCI small fragment tasks of the OAEI LargeBio Track. They have large-scale ontologies and high quality gold standards created by domain experts. Table 1 summarizes the numbers of classes in source (SRC) and target (TGT) ontologies, and the numbers of reference mappings. "Refs (=)" refers to the reference mappings to be considered, while "Refs (?)" refers to the reference mappings that will cause logical inconsistency after alignment and are ignored as suggested by OAEI. We also consider an extended task of FMA-SNOMED, denoted as FMA-SNOMED+, where the target ontology is extended by introducing the labels from the latest version of SNOMED.[6] This is because the LargeBio SNOMED is many years out of date, and the naming scheme in the newly released SNOMED has changed and many more class labels have been added. We adopt the following strategy to construct SNOMED+: for each class c in SNOMED, we extract its labels $\Omega(c)$ and for each label ω in $\Omega(c)$, we search for classes in the original SNOMED that have ω as an alias; we then add all the labels of the searched classes to the LargeBio SNOMED for SNOMED+. We also use these additional labels to construct the complementary corpus for the FMA-SNOMED task. The key difference is that they are used for fine-tuning alone on the FMA-SNOMED task but for both fine-tuning and prediction on the FMA-SNOMED+ task.

Evaluation Metrics We evaluate all the systems on Precision (P), Recall (R), and Macro-F1 (F1), defined as:

$$P = \frac{|\mathcal{M}_{out} \cap \mathcal{M}_= \backslash \mathcal{M}_?|}{|\mathcal{M}_{out} \backslash \mathcal{M}_?|}, R = \frac{|\mathcal{M}_{out} \cap \mathcal{M}_= \backslash \mathcal{M}_?|}{|\mathcal{M}_= \backslash \mathcal{M}_?|}$$

and $F1 = 2PR/(P + R)$, where $\mathcal{M}_{out}$ is the system's output mappings, $\mathcal{M}_=$ and $\mathcal{M}_?$ refer to reference mappings to be considered (Refs (=)) and ignored (Refs (?)), respectively. In the unsupervised setting, we divide $\mathcal{M}_=$ into $\mathcal{M}_{val}$ (10%) and $\mathcal{M}_{test}$ (90%); and in the semi-supervised setting, we divide $\mathcal{M}_=$ into $\mathcal{M}_{train}$ (20%), $\mathcal{M}_{val}$ (10%) and $\mathcal{M}_{test}$ (70%). When computing the metrics on the hold-out validation or test set, we should regard reference mappings that are not in this set as neither positive nor negative (i.e., as ignored mappings). For example, during validation, we add the mappings from $\mathcal{M}_{train}$ (if semi-supervised) and $\mathcal{M}_{test}$ (for both settings) into $\mathcal{M}_?$ when calculating the metrics.

---

[6]The version of 20210131 is available at: https://www.nlm.nih.gov/healthit/snomedct/index.html.

such as parents and children are likely to be matched. This is referred to as the *locality principle* which is assumed in many ontology engineering tasks (Grau et al. 2007; Jiménez-Ruiz et al. 2020). BERTMap utilizes this principle to discover new mappings from those highly scored mappings with an *iterative mapping extension* algorithm (see Algorithm 1). Note that this algorithm only preserves extended mappings that are not previously seen (in $\mathcal{M}$ and $\mathcal{M}_{ex}$) and have scores $\geq \kappa$ (Line 10 - 12), i.e. the extension threshold. Moreover, although κ is a hyperparameter, the empirical evidence shows that the results are insensitive to κ, and thus we set it to a fixed value $\kappa = 0.9$. Finally, the algorithm terminates iteration when no new mappings can be found.

Mapping Repair Mapping repair removes mappings that will lead to logical conflicts after integrating two ontologies. A "perfect repair" (a.k.a. a *diagnosis*) refers to removing a minimal number of mappings to achieve logical coherence. However, computing a diagnosis is usually time-consuming, and there may be no unique solution. To address this, Jiménez-Ruiz et al. (2013) proposes a propositional logic-based repair method that can efficiently compute an *approximate* repair R which ensures that: *(i)* R is a subset of the diagnosis (so that there is no sacrifice of correct mappings); *(ii)* only a small number of unsatisfiable classes remain. Mapping repair is commonly used in classic OM systems, but rarely considered in machine learning-based approaches. In this work, we adopt the repair tool developed by Jiménez-Ruiz et al. (2013).

Note that mapping extension and repair can consistently improve the performance without excessive time cost, because the former only needs to handle mappings of high prediction scores and the later adopts an efficient repair al-

BERTMap Settings We set up various BERTMap settings considering *(i)* being unsupervised (without co) or semi-supervised ($+co$), *(ii)* including the identity synonyms ($+ids$), *(iii)* being augmented with a complementary corpus ($+cp$), and *(iv)* applying mapping extension (ex) and repair (rp). In fine-tuning, the semi-supervised setting takes all the label pairs extracted from both within the input ontologies and $\mathcal{M}_{train}$ as training data, label pairs from $\mathcal{M}_{val}$ as validation data and label pairs from $\mathcal{M}_{test}$ as test data, while the unsupervised setting partitions all the label pairs extracted from within the input ontologies into 80% for training and 20% for validation. Note that the *the validation in fine-tuning* is different from the *mapping validation* which uses $\mathcal{M}_{val}$ because the former concerns the performance of the BERT classifier while the latter concerns selecting the best hyperparameters for determining output mappings.

Besides, we set the positive-negative sample ratio to $1:4$. Namely, we sample 4 non-synonyms for each synonym in co, and 2 soft and 2 hard non-synonyms for each synonym in other corpora. We use Bio-Clinical BERT, which has been pre-trained on biomedical and clinical domain corpora (Alsentzer et al. 2019). The BERT model is fine-tuned for 3 epochs with a batch size of 32, and evaluated on the validation set for every 0.1 epoch, through which the best checkpoint on the cross-entropy loss is selected for prediction. The cut-off of sub-word inverted index-based candidate selection is set to 200. Our implementation uses *(i)* `owlready2`[7] for ontology processing and *(ii)* `transformers`[8] for BERT. The training uses a single GTX 1080Ti GPU.

After fine-tuning, we perform a 2-step mapping validation using $\mathcal{M}_{val}$ as follows: we first validate the scored mappings from prediction and obtain the best $\{\tau, \lambda\}$; we then extend the mappings by Algorithm 1 and validate the extended mappings and obtain another best mapping filtering threshold λ. Interestingly, in all our BERTMap experiment settings, we find the best λ obtained in the first step always coincides with the best λ obtained in the second step. This demonstrates the robustness of our mapping extension algorithm. After validation, we repair and ouput the mappings. Note that we also test BERTMap without extension and repair, and in this case, we skip the second mapping validation step and output mappings with scores $\geq \lambda$.

Baselines We compare BERTMap with various baselines as follows: *(i)* *String-matching* as defined in the Mapping Score Computation; *(ii)* *Edit-similarity*, which computes the maximum normalized edit similarity between the labels of two classes as their mapping score (note that *(i)* is a special case of *(ii)*); *(iii)* *LogMap* and *AML*, which are the leading systems in many OAEI tracks and other tasks; *(iv)* *LogMapLt*, the lexical matching part of LogMap; *(v)* *LogMap-ML**, which is a variant of LogMap-ML (Chen et al. 2021b) using no branch conflicts but only LogMap anchor mappings for extracting samples for training, where Word2Vec is used to embed the class label and a Siamese Neural Network with Multilayer Perception is used as the classifier. Note that *(i)* and *(ii)* are our internal baselines, and

we set up the same candidate selection and hyperparameter search procedure for them as for BERTMap; whereas *(iii)* to *(v)* are external systems with default implementations. Note by comparing to LogMap and AML, we actually have several indirect baselines that have participated in the LargeBio Track (e.g., ALOD2Vec (Portisch and Paulheim 2018) and Wiktionary (Portisch, Hladik, and Paulheim 2019)).

Results

The results together with the corresponding hyperparameter settings are shown in Tables 2, 3 and 4, where 90% (resp. 70%) Test Mappings refer to the results measured on $\mathcal{M}_{test}$ of the unsupervised (resp. semi-supervised) setting. To fairly compare the unsupervised and semi-supervised settings, we report the results on both 90% and 70% Test Mappings for the unsupervised setting.

The overall results show that BERTMap can achieve higher F1 score than all the baselines on the FMA-SNOMED and FMA-SNOMED+ tasks, but its F1 score is lower than LogMap and AML on the FMA-NCI task. On the FMA-SNOMED task, the unsupervised BERTMap can surpass AML (resp. LogMap) by 1.4% (resp. 4.2%) in F1, while the semi-supervised BERTMap can exceed AML (resp. LogMap) by 3.0% (resp. 5.4%). The corresponding rates become 2.5% (resp. 1.8%) and 3.3% (resp. 2.7%) on the FMA-SNOMED+ task. On the FMA-NCI task, the best F1 score of the unsupervised BERTMap is worse than AML (resp. LogMap) by 2.5% (resp. 2.6%), and the best F1 score of the semi-supervised BERTMap is worse than AML (resp. LogMap) by 2.3% (resp. 2.3%). Note that BERTMap without ex or rp consistently outperforms LogMapLt on all the tasks. This suggests that with a more suitable mapping refinement strategy, BERTMap is likely to outperform LogMap on the FMA-NCI task as well. BERTMap can also significantly outperform the machine learning-based baseline LogMap-ML* on all the three tasks. This is because LogMap-ML* relies on LogMap and heuristic rules to extract high quality samples (anchor mappings) for training, but this strategy is not effective on our data. In contrast, BERTMap primarily relies on unsupervised data (synonyms and non-synonyms) to fine-tune the BERT model.

By comparing different BERTMap settings, we have the following observations. First, the semi-supervised setting ($+co$) is generally better than the unsupervised setting (without co), implying that BERTMap can effectively learn from given mappings. Second, complementary corpus is helpful especially when the task-involved ontologies are deficient in class labels — on the FMA-SNOMED task, BERTMap with the complementary corpus ($+cp$) attains a higher F1 score than string-matching, edit-similarity, LogMapLt and LogMap-ML* baselines, all of which rely on class labels from within the input ontologies, by around 50%. Third, considering the identity synonyms ($+ids$) may slightly improve the performance or make no difference. Finally, mapping extension and repair can consistently boost the performance, but not by much, possibly because it is hard to improve given that BERTMap's prediction part has already achieved high performance.

It is interesting to notice that BERTMap is robust to hy-

System	$\{\tau, \lambda\}$	90% Test Mappings			70% Test Mappings		
		Precision	Recall	Macro-F1	Precision	Recall	Macro-F1
io	(tgt2src, 0.999)	0.705	0.240	0.359	0.649	0.239	0.350
io+ids	(tgt2src, 0.999)	0.835	0.347	0.490	0.797	0.346	0.483
io+cp	(src2tgt, 0.999)	0.917	0.750	0.825	0.895	0.748	0.815
io+ids+cp	(src2tgt, 0.999)	0.910	0.758	0.827	0.887	0.755	0.816
io+ids+cp (ex)	(src2tgt, 0.999)	0.896	0.771	0.829	0.869	0.771	0.817
io+ids+cp (ex+rp)	(src2tgt, 0.999)	0.905	0.771	**0.833**	0.881	0.771	0.822
io+co	(src2tgt, 0.997)	NA	NA	NA	0.937	0.564	0.704
io+co+ids	(src2tgt, 0.999)	NA	NA	NA	0.850	0.714	0.776
io+co+cp	(src2tgt, 0.999)	NA	NA	NA	0.880	0.779	0.826
io+co+ids+cp	(src2tgt, 0.999)	NA	NA	NA	0.899	0.774	0.832
io+co+ids+cp (ex)	(src2tgt, 0.999)	NA	NA	NA	0.882	0.787	0.832
io+co+ids+cp (ex+rp)	(src2tgt, 0.999)	NA	NA	NA	0.892	0.786	**0.836**
string-match	(combined, 1.000)	0.987	0.194	0.324	0.983	0.192	0.321
edit-similarity	(combined, 0.920)	0.971	0.209	0.343	0.963	0.208	0.343
LogMapLt	NA	0.965	0.206	0.339	0.956	0.204	0.336
LogMap	NA	0.935	0.685	0.791	0.918	0.681	0.782
AML	NA	0.892	0.757	0.819	0.865	0.754	0.806
LogMap-ML[*]	NA	0.944	0.205	0.337	0.928	0.208	0.340

Table 2: Results of BERTMap under different settings and baselines on the FMA-SNOMED task.

System	$\{\tau, \lambda\}$	90% Test Mappings			70% Test Mappings		
		Precision	Recall	Macro-F1	Precision	Recall	Macro-F1
io	(src2tgt, 0.999)	0.930	0.836	0.880	0.911	0.834	0.871
io+ids	(src2tgt, 0.999)	0.926	0.834	0.878	0.906	0.832	0.868
io+ids (ex)	(src2tgt, 0.999)	0.916	0.852	0.883	0.894	0.851	0.872
io+ids (ex+rp)	(src2tgt, 0.999)	0.924	0.851	**0.886**	0.905	0.851	0.877
io+co	(src2tgt, 0.999)	NA	NA	NA	0.913	0.841	0.875
io+co+ids	(src2tgt, 0.999)	NA	NA	NA	0.913	0.836	0.873
io+co+ids (ex)	(src2tgt, 0.999)	NA	NA	NA	0.899	0.852	0.875
io+co+ids (ex+rp)	(src2tgt, 0.999)	NA	NA	NA	0.908	0.852	**0.879**
string-match	(src2tgt, 1.000)	0.978	0.672	0.797	0.972	0.665	0.790
edit-similarity	(src2tgt, 0.930)	0.978	0.728	0.834	0.972	0.724	0.830
LogMapLt	NA	0.953	0.717	0.819	0.940	0.709	0.808
LogMap	NA	0.869	0.867	0.868	0.838	0.868	0.852
AML	NA	0.895	0.829	0.861	0.868	0.825	0.846
LogMap-ML[*]	NA	0.955	0.684	0.797	0.942	0.700	0.803

Table 3: Results of BERTMap under different settings and baselines on the FMA-SNOMED+ task.

System	$\{\tau, \lambda\}$	90% Test Mappings			70% Test Mappings		
		Precision	Recall	Macro-F1	Precision	Recall	Macro-F1
io	(src2tgt, 0.999)	0.930	0.847	0.887	0.912	0.851	0.880
io+ids	(src2tgt, 0.999)	0.936	0.842	0.887	0.920	0.845	0.881
io+ids (ex)	(src2tgt, 0.999)	0.926	0.852	0.888	0.907	0.854	0.880
io+ids (ex+rp)	(src2tgt, 0.999)	0.938	0.852	0.893	0.922	0.854	0.887
io+co	(src2tgt, 0.999)	NA	NA	NA	0.939	0.838	0.886
io+co+ids	(src2tgt, 0.999)	NA	NA	NA	0.961	0.805	0.876
io+co+ids (ex)	(src2tgt, 0.999)	NA	NA	NA	0.955	0.813	0.879
io+co+ids (ex+rp)	(src2tgt, 0.999)	NA	NA	NA	0.959	0.813	0.880
string-match	(tgt2src, 1.000)	0.980	0.727	0.835	0.975	0.733	0.837
edit-similarity	(src2tgt, 0.900)	0.976	0.768	0.860	0.970	0.774	0.861
LogMapLt	NA	0.963	0.815	0.883	0.953	0.812	0.877
LogMap	NA	0.938	0.900	**0.919**	0.922	0.897	**0.909**
AML	NA	0.936	0.900	0.918	0.919	0.898	**0.909**
LogMap-ML[*]	NA	0.968	0.715	0.822	0.959	0.714	0.818

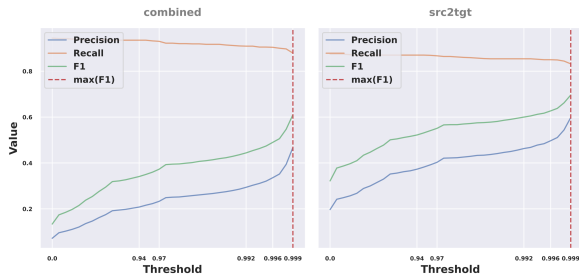Table 4: Results of BERTMap systems under different settings and baselines on the FMA-NCI task.

Figure 2: Validation results of BERTMap ($io + co + ids$) on the FMA-SNOMED+ task with mapping score threshold λ ranging from 0 to 1. The left and right plots correspond to $\tau = \mathtt{combined}$ and $\tau = \mathtt{src2tgt}$, respectively.

FMA Class	SNOMED Class
Third_cervical_spinal_ganglion	C3_spinal_ganglion
Deep_posterior_sacrococcygeal ligament	Structure_of_deep_dorsal _sacrococcygeal_ligament
Wall_of_smooth_endoplasmic _reticulum	Agranular_endoplasmic _reticulum_membrane

Table 5: Typical examples of reference mappings that are predicted by BERTMap but not by LogMap or AML.

perparameter selection; most of its settings lead to the same best hyperparameters (i.e. $\tau = \mathtt{src2tgt}$ and $\lambda = 0.999\}$) on the validation set, $\mathcal{M}_{val}$. To further investigate this phenomenon, we visualize the validation process by presenting the plots of evaluation metrics against λ in Figure 2, where we can see that as λ increases, Precision increases significantly while Recall drops only slightly — thus F1 increases and attains the maximum at $\lambda = 0.999$. This observation is consistent for all BERTMap models in this paper[9].

In Table 5, we present some examples of reference mappings that are retrieved by BERTMap but not by LogMap or AML. We can clearly see that, the BERT classifier captures the implicit connection between *"third cervical"* and *"C3"* in the first example, *"posteior"* and *"dorsal"* in the second example, as well as *"wall"* and *"membrane"* in the third example. This demonstrates the strength of contextual embeddings over the traditional lexical matching.

Related Work

Classic OM systems are often based on lexical matching, structure matching and logical inference (Otero-Cerdeira, Rodríguez-Martínez, and Gómez-Rodríguez 2015). For example, LogMap (Jiménez-Ruiz and Cuenca Grau 2011) uses a lexical index to compute anchor mappings, then alternates between mapping extension that utilizes ontology structure, and mapping repair that utilizes logical reasoning; whereas AML (Faria et al. 2013) mixes several strategies to calculate lexical matching scores, followed by mapping extension and repair. Although these systems have proven quite effective,

[9]See appendix at: https://arxiv.org/abs/2112.02682.

their lexical matching only utilizes the surface form of texts and ignores word semantics. BERTMap employs a similar architecture but utilizes BERT so that textual semantics and contexts are considered in mapping computation.

Recent supervised learning-based OM approaches mainly focus on constructing class embeddings or extracting features. Nkisi-Orji et al. (2018) uses hand-crafted features such as string similarities together with Word2Vec; OntoEmma (Wang et al. 2018) relies on both hand-crafted features and word context features learned by a complex network; LogMap-ML (Chen et al. 2021b) utilizes path contexts and ontology tailored word embeddings by OWL2Vec* (Chen et al. 2021a) (or just word embeddings); VeeAlign (Iyer, Agarwal, and Kumar 2020) proposes "dual attention" for class embeddings. However, these approaches often heavily depend on complicated feature engineering and/or complex neural networks. More importantly, they need a significant number of high quality labeled mappings for training which are often not available and costly to manually annotate. Although some solutions such as distant supervision (Chen et al. 2021b) and sample transfer (Nkisi-Orji et al. 2018) have been investigated, the sample quality often varies and limits their performance. Unsupervised learning approaches such as ERSOM (Xiang et al. 2015) and DeepAlignment (Kolyvakis, Kalousis, and Kiritsis 2018) have also been studied. They attempt to refine word embeddings by, e.g., counter-fitting to directly compute class similarity. However, they do not consider word contexts.

Neutel and Boer (2021) have presented a preliminary OM investigation using BERT. Their work considered two relatively naive approaches: *(i)* encoding classes with pre-trained BERT's token embeddings and calculating their cosine similarity; *(ii)* fine-tuning class embeddings with the SentenceBERT (Reimers and Gurevych 2019) architecture, which relies on a large number of given mappings. We have implemented *(i)* and found it to perform much worse than string-matching on our tasks; moreover, according to their evaluation, method *(ii)* has much lower mean reciprocal rank score than the non-contextual word embedding model, FastText (Bojanowski et al. 2017), although it has higher coverage. Furthermore, their evaluation data used have no gold standards with Precision, Recall and F1 not computed.

Conclusion and Future Work

In this paper, we propose a novel, general and practical OM system, BERTMap, which exploits the textual, structural and logical information of ontologies. The backbone of BERTMap is its predictor, which utilizes the contextual embedding model, BERT, to learn word semantics and contexts effectively, and computes mapping scores with the aid of sub-word inverted indices. The mapping extension and repair modules further improve the recall and precision, respectively. BERTMap works well with just the to-be-aligned ontologies and can be further improved by given mappings and/or complementary sources. In future, we will evaluate BERTMap with more large-scale (industrial) data. We will also consider e.g., BERT-based ontology embedding for more robust mapping prediction, and more paradigms for integrating mapping prediction, extension and repair.

Acknowledgments

References

Alsentzer, E.; Murphy, J.; Boag, W.; Weng, W.-H.; Jindi, D.; Naumann, T.; and McDermott, M. 2019. Publicly Available Clinical BERT Embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, 72–78.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5: 135–146.

Chen, J.; Hu, P.; Jimenez-Ruiz, E.; Holter, O. M.; Antonyrajah, D.; and Horrocks, I. 2021a. OWL2Vec*: Embedding of OWL ontologies. *Machine Learning*, 1–33.

Chen, J.; Jiménez-Ruiz, E.; Horrocks, I.; Antonyrajah, D.; Hadian, A.; and Lee, J. 2021b. Augmenting ontology alignment by semantic embedding and distant supervision. In *European Semantic Web Conference*, 392–408. Springer.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*, 4171–4186.

Faria, D.; Pesquita, C.; Santos, E.; Palmonari, M.; Cruz, I. F.; and Couto, F. M. 2013. The AgreementMakerLight Ontology Matching System. In Meersman, R.; Panetto, H.; Dillon, T.; Eder, J.; Bellahsene, Z.; Ritter, N.; De Leenheer, P.; and Dou, D., eds., *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, 527–541. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-41030-7.

Grau, B. C.; Horrocks, I.; Kazakov, Y.; and Sattler, U. 2007. A Logical Framework for Modularity of Ontologies. In *IJCAI*.

Horrocks, I.; Chen, J.; and Lee, J. 2020. Tool Support for Ontology Design and Quality Assurance.

Iyer, V.; Agarwal, A.; and Kumar, H. 2020. VeeAlign: a supervised deep learning approach to ontology alignment. In *OM@ISWC*.

Jiménez-Ruiz, E.; Agibetov, A.; Chen, J.; Samwald, M.; and Cross, V. V. 2020. Dividing the Ontology Alignment Task with Semantic Embeddings and Logic-based Modules. *ArXiv*, abs/2003.05370.

Jiménez-Ruiz, E.; and Cuenca Grau, B. 2011. LogMap: Logic-Based and Scalable Ontology Matching. In Aroyo, L.; Welty, C.; Alani, H.; Taylor, J.; Bernstein, A.; Kagal, L.; Noy, N.; and Blomqvist, E., eds., *The Semantic Web – ISWC 2011*, 273–288. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-25073-6.

Jiménez-Ruiz, E.; Meilicke, C.; Grau, B. C.; and Horrocks, I. 2013. Evaluating Mapping Repair Systems with Large Biomedical Ontologies. In *Description Logics*.

Kolyvakis, P.; Kalousis, A.; and Kiritsis, D. 2018. DeepAlignment: Unsupervised Ontology Matching with Refined Word Vectors. In *Proceedings of NAACL-HLT*, 787–798.

Loshchilov, I.; and Hutter, F. 2017. Fixing Weight Decay Regularization in Adam. *ArXiv*, abs/1711.05101.

Mikolov, T.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

Neutel, S.; and Boer, M. D. 2021. Towards Automatic Ontology Alignment using BERT. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*.

Nkisi-Orji, I.; Wiratunga, N.; Massie, S.; Hui, K.-Y.; and Heaven, R. 2018. Ontology alignment based on word embedding and random forest classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 557–572. Springer.

Otero-Cerdeira, L.; Rodríguez-Martínez, F. J.; and Gómez-Rodríguez, A. 2015. Ontology matching: A literature review. *Expert Systems with Applications*, 42(2): 949–971.

Portisch, J.; Hladik, M.; and Paulheim, H. 2019. Wiktionary Matcher. In *OM@ISWC*.

Portisch, J.; and Paulheim, H. 2018. ALOD2Vec matcher. In *OM@ISWC*.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *ArXiv*, abs/1908.10084.

Shvaiko, P.; and Euzenat, J. 2013. Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1): 158–176.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wang, L.; Bhagavatula, C.; Neumann, M.; Lo, K.; Wilhelm, C.; and Ammar, W. 2018. Ontology alignment in the biomedical domain using entity definitions and context. In *Proceedings of the BioNLP 2018 workshop*, 47–55.

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.; and Dean, J. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*.

Xiang, C.; Jiang, T.; Chang, B.; and Sui, Z. 2015. Ersom: A structural ontology matching approach using automatically learned entity representation. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2419–2429.