Ontology Matching with Pre-trained Language Model

Yuan He

St Hugh's College

University of Oxford

Supervisors: Prof. Ian Horrocks, Prof. Bernardo Cuenca Grau

Transfer of Status Report
Doctor of Philosophy in Computer Science

Michaelmas 2021

Contents

List of Figures

Chapter 1

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 [9, 29]. 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 [13] and AgreementMakerLight (AML) [8] which still demonstrate the 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 [16] and OntoEmma [32] 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 [21], 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 [7] can learn robust

contextual word embeddings, and usually require only moderate training resources for fine-tuning. Such models are typically pre-trained on gigantic corpora to learn general background knowledge, and then released to the community such that people can download the model and fine-tune it on customized downstream tasks. The process of adjusting the parameters of the pre-trained model according to the downstream objective function is referred to as *fine-tuning*, which has been proven rather effective in many Natural Language Processing (NLP) tasks, but has not yet been sufficiently investigated in OM.

In this report, 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 1.1, BERTMap includes the following main steps:

- **Corpus Construction**, where synonym and non-synonym pairs from various ontology-related sources are extracted;
- **Fine-tuning**, where a suitable pre-trained BERT model is selected and fine-tuned on the corpora constructed in the previous step;
- **Mapping Prediction**, where mapping candidates are first selected based on sub-word inverted indices and then predicted by the fine-tuned BERT classifier;
- **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 OAEI Large BioMed Track[2] (LargeBio), which consists of three well-known biomedical ontologies (but of old versions), namely the Foundational Model of Anatomy (FMA), SNOMED CT, and the National Cancer Institute Thesaurus (NCI). These ontologies are abundant in concepts and semantics. The ground truth mappings are constructed based on UMLS Metathesaurus[3], which is currently the most comprehensive crowd-source effort for integrating independently-developed medical thesauri and ontologies. We specifically choose the FMA-SNOMED and FMA-NCI tasks and conduct in-depth ablation study to examine BERTMap and other benchmarks. We also consider an extended task of FMA-SNOMED (denoted as FMA-SNOMED+) where the more complete labels from

[1]Our source codes and data are available at: `https://github.com/KRR-Oxford/BERTMap`.
[2]`http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/`
[3]`https://www.nlm.nih.gov/research/umls/index.html`

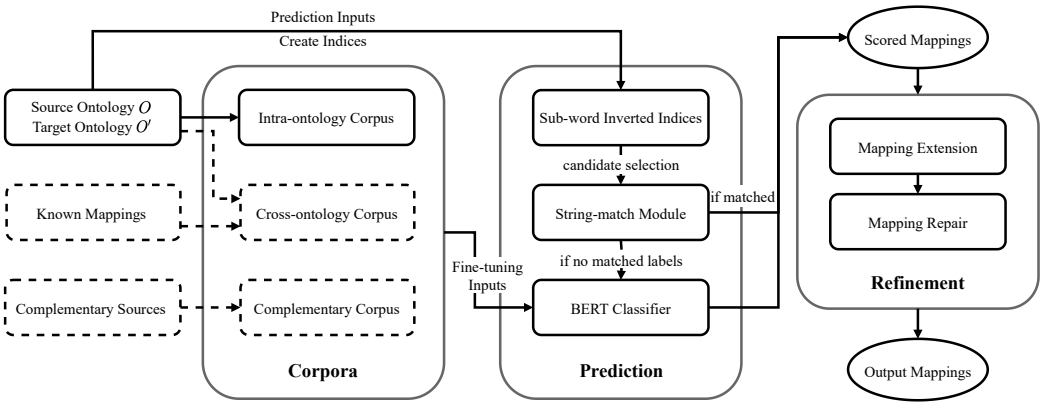

Figure 1.1: Illustration of BERTMap system.

the up-to-date SNOMED ontology are reintroduced to the LargeBio SNOMED. We will explain the reason for setting this additional task in the experiment section (see Chapter 4). Our results on these biomedical ontology alignment tasks demonstrate that BERTMap can often outperform the state-of-the-art systems LogMap and AML.

The outline of the rest of the report is presented below:

- In Chapter 2, we will introduce the task definition of OM and some relevant concepts, background knowledge concerning BERT, and literature related to OM.
- In Chapter 3, we will explain BERTMap in full technical details.
- In Chapter 4, we will illustrate how we evaluate BERTMap and other baseline systems (including the state-of-the-art ones) on three biomedical ontology alignment tasks.
- In Chapter 5, we will discuss principles of OM systems, whether or not mapping repair is necessary, and conclude our work with a qualitative analysis of BERTMap from several aspects.
- In Chapter 6, we will discuss our future work as a series of projects towards data integration using OM-PLM (Ontology Matching with Pre-trained Language Modelling).

Chapter 2

Background

2.1 Ontology Alignment

An ontology is typically defined as an explicit specification of a conceptualization. Its main components are *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 identifying 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 (a.k.a., mapping value); we then extend and repair the scored mappings to output determined mappings.

To be more specific, we intend to first establish a mapping scoring function:

$$s_{map} : C \times C' \to [0, 1]$$

that takes two classes as input, and computes the equivalence score between them. This function can be constructed either using heuristics and rules or can be learnt through optimizing a customized objective function (i.e., backpropagation [28]). We also need a candidate selection function, defined as:

$$f_{sel} : C \times \mathcal{P}(C') \to \mathcal{P}(C')$$

where $\mathcal{P}(\cdot)$ refers to the *power set*[1]. Note that the simplest method of candidate selection is to traverse the whole target class space, resulting in a quadratic time complexity. In BERTMap, we propose a candidate selection algorithm that requires

[1]The set of all the subsets.

only linear time. The process of selecting target candidate classes and predict a mapping for each source class is referred to as *mapping prediction*.

Moreover, as the last step of the sequential procedure, we also consider *mapping refinement*, which consists of *mapping extension* and *mapping repair*. The extension function aims to discover more mappings from existing (predicted) mappings by utilizing the graphical information of input ontologies. In this work, we consider the following principle:

Principle 2.1.1 (**Locality Principle**). *Suppose there exists a mapping between $c \in C$ and $c' \in C'$, classes that are semantically related to c are likely to be mapped to those semantically related to c'.*

For example, if c is equivalent to c' through a correct mapping, their parents and children are likely to be matched, respectively. Details of how to apply the locality principle to establish the extension module will be presented in Chapter 3.

Mapping repair aims to detect and delete mappings that will cause logical errors (*incoherence*) after integrating two ontologies. We borrow the definition of mapping repair from [15]:

Definition 2.1.2 (**Repair**). *If a set of mappings $\mathcal{M}$ between ontologies O and O' leads to logical incoherence, then $\mathcal{R} \subseteq \mathcal{M}$ is a mapping repair if $\mathcal{M} \setminus \mathcal{R}$ is coherent w.r.t. O and O'.*

A trivial example of mapping repair is to let $\mathcal{R} = \mathcal{M}$ such that the mapping set becomes empty. But what we really need is to remove a minimal number of mappings to achieve coherence. Such "perfect repair", a.k.a. *diagnosis*, is defined as:

Definition 2.1.3 (**Diagnosis**). *If $\mathcal{R}$ is a repair for a set of mappings $\mathcal{M}$ w.r.t. ontologies O and O', then $\mathcal{R}$ is a diagnosis if $R' \subset R$ is not a repair for $\mathcal{M}$ w.r.t. O and O'.*

Nevertheless, computing a diagnosis confronts the scalability issue when the number of unsatisfiability increases. In practice, mapping repair systems often *approximate* the diagnosis using partial reasoning techniques. Such approximation does not guarantee to remove all the incoherent mappings but to a large extent preserve most of the coherent mappings. In BERTMap, we utilize the mapping repair tool developed in [15] as an optional step of mapping refinement.

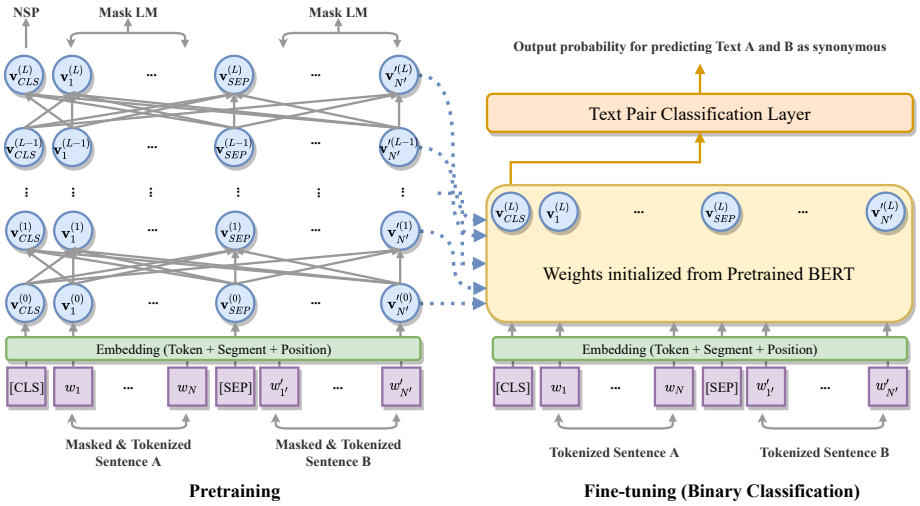

Figure 2.1: Illustration of BERT pre-training (left) and fine-tuning (right).

2.2 BERT: Pre-training and Fine-tuning

BERT is a contextual language representation model built on bidirectional transformer encoders [31]. Its framework involves *pre-training* and *fine-tuning*. In pre-training (see Figure 2.1, left), 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. We present the outputs at layer l in the following equation:

$$f_{bert}(\mathbf{x}, l) = (\mathbf{v}_{CLS}^{(l)}, \mathbf{v}_1^{(l)}, ..., \mathbf{v}_N^{(l)}, \mathbf{v}_{SEP}^{(l)}, \mathbf{v}_1'^{(l)}, ..., \mathbf{v}_{N'}'^{(l)}) \in \mathbb{R}^{(N+N'+2)\times d} \tag{2.1}$$

where $\mathbf{x}$ is the initial embedding of the input sequence, $(\mathbf{v}_i^{(l)})_{i=1,...N}$ are computed embeddings of the i^{th} token in the input sentence A at layer l, $(\mathbf{v}_i'^{(l)})_{j=1,...N'}$ are computed embeddings of the j^{th} token in the input sentence B at layer l, $\mathbf{v}_{CLS}^{(l)}$ and $\mathbf{v}_{SEP}^{(l)}$ are embeddings of the `[CLS]` and `[SEP]` tokens at layer l, respectively. Note that each token embedding is of the dimension d (e.g. $d = 768$ for the `bert-base`[2]

[2]See a list of available pre-trained BERT models at: `https://huggingface.co/transformers/pretrained_models.html`.

model).

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 such as Word2Vec [21], which assign each token in the vocabulary 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 (probably the money bank) in the corpus.

To directly use the contextual embeddings generated by the pre-trained BERT, the author suggested to either concatenate or average the last-few-layer outputs. Nevertheless, it argues in [36] that we should use the second-to-last layer outputs because they reach the sweet point between learning and generalization. Details of BERT probing, and even the whole branch of studying how BERT is working and what BERT can do (referred to as "BERTology") can be seen in [27] as a good starting point.

In fine-tuning, pre-trained BERT is attached to customized downstream layers and takes as input either one sentence or two sentences according to specific tasks. We present some examples as follows:

(i) In sentiment analysis, it takes one sentence as input and outputs the category of sentiment.

(ii) In paraphrasing task, it takes two sentences as input and tries to learn semantic similarity between the input sentence pair, and outputs the corresponding probability score.

(iii) In question answering, it takes two sentences as input but one states the question and the other represents the context of the question. The output can be, for example, a simple answer with constrained patterns, or a complex answer associated to some logical representation graph.

Fine-tuning typically necessitates only a few epochs and a moderate number of samples for training. In the right part of Figure 2.1, we present an example of fine-tuning on a binary classification task where it takes a sentence pair as input, and outputs the probability that the input sentences are synonymous.

2.3 Related Work

2.3.1 Traditional OM Systems

Classic OM systems are often based on lexical matching, structure matching and logical inference [25]. We exemplify two traditional OM systems, LogMap[3] [13] and AML [8] with the state-of-the-art performance.

2.3.1.1 LogMap

The first step of LogMap pipeline is to construct lexical indices for input ontologies with word form variations considered, and compute their intersection, i.e., to check if any of the lexical entries contain the same strings. Classes that are values of the overlapped entries are then matched and each such cross-ontology class pair is deemed as an *anchor mapping*, with a confidence score that also considers neighbourhood similarity. Based on these anchor mappings, LogMap alternates between mapping extension (discovery) that utilizes structural indexation, and mapping repair that utilizes logical reasoning to expand the mapping set while preserving logical consistency. LogMap has several variants including:

(i) LogMapLt, which is the lexical matching part of LogMap;

(ii) LogMapBio, which uses biomedical background knowledge to enrich its indexation.

(iii) LogMap-ML [6], which is a machine learning extension of LogMap where the anchor mappings are used for downstream distant supervision of a Siamese network, together with an ontology embedding model, OWL2Vec* [5].

These systems have been consistently developed over the past decade, and in particular, participate the OAEI tracks for many years with leading performance, so they are naturally strong baselines considered in our evaluation.

2.3.1.2 AgreementMakerLight

Unlike LogMap, AgreementMakerLight (AML) is not an iterative process; instead, it focuses more on mixed matching strategies. AML also adopts a lexical matcher similar to LogMap, but the matching module is further enhanced by three other matchers including:

[3]LogMap has recently won the ISWC 2021 Ten-Year-Reward, see `https://twitter.com/iswc_conf/status/1452974140068188167`.

(i) Mediating Matcher, which utilizes a meditating ontology as an intermedidate between the input ontologies to build "bridge" alignments;

(ii) Word Matcher, which computes class simlarities based on several string simlarity-based algorithms;

(iii) Parametric String Matcher, which serves as the secondary matcher of AML (only used in extension mode) for non-literal comparisons.

AML also supports mapping extension and repair, but they are not as important as in LogMap.

Although such system architectures, which integrate structure matching and logic-based repair, have proven quite effective, their lexical matching, such as the lexical index and edit distance metrics, only utilizes the surface form of texts and ignores word semantics. Our proposed OM solution, BERTMap, employs a similar architecture, but utilizes fine-tuned BERT so that textual semantics and contexts are considered in mapping computation.

2.3.2 Machine Learning-based OM Systems

The machine learning-based OM systems can be categorized as *supervised* or *unsupervised*, where the former requires annotated mappings for training but the latter does not. In this section, we illustrate OM solutions under these two different settings.

2.3.2.1 Supervised OM Approaches

A fundamental challenge of apply supervised learning scheme in OM is the *extreme imbalance* between positive and negative samples, i.e., there are several orders of magnitude less correct mappings than incorrect ones (which can be easily extracted from randomly aligned class pairs). To address this, the supervised OM solutions often rely on *silver data* (automatically annotated by some heuristics rather than human-annotated) and/or complex feature engineering.

OntoEmma [32] is an typical example of supervised OM solution that relies on both hand-crafted and automatically learnt features. It attempts to learn class representations with textual context features considered through a highly complex network that uses character-level CNN and Word2Vec embeddings to initialize text tokens used to represent a class and further improve the embeddings through bi-directional LSTMs; the learnt class embeddings are then sent to a Siamese network for computing a similarity score. Note that the text features consider "contexts" which are sentences that

mention the associated concept in some external materials. Frustratingly, the model performance is much worse than AML on a LargeBio task, and most of the gains are owing to hand-crafted features rather than the learnt complex features.

Other examples like [24] uses hand-crafted features such as string similarities together with Word2Vec; LogMap-ML [6] utilizes path contexts and ontology embeddings from OWL2Vec* [5]; VeeAlign [10] proposes "dual attention" (path and node level) in its model to enhance the class embeddings.

However, all of these approaches depend on complicated feature engineering and/or complex neural network architectures. 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 [6] and sample transfer [24] have been investigated, the sample quality often varies and limits their performance.

2.3.2.2 Unsupervised OM Approaches

Unsupervised learning approaches such as ERSOM [35] and DeepAlignment [16] have also been studied in OM. ERSOM attempts to use an auto-encoder (a network that learns how to map the input to itself) to learn class embeddings through a combination of entity descriptions; while DeepAlignment adopts *counter-fitting* [22], a refinement technique that utilizes synonyms and non-synonyms to adjust word embeddings. Both of these approaches try to encode a class without requiring annotaed mappings, but such representations are relatively naive because they do not take textual contexts into consideration.

We will soon see our proposed model, BERTMap, is mainly an unsupervised model that requires no annotated data, but can be further enhanced by a small portion of known mappings under the semi-supervised setting. BERTMap is better than the machine learning-based OM solutions discussed above because of its flexible learning pipeline and its incorporation of both text-level and ontology-level contexts.

2.3.3 Preliminary OM Work on Using BERT

Neutel and Boer [23] have already presented a preliminary investigation of the use of BERT in OM. 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 [26] 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 paper, method *(ii)* has much lower mean reciprocal rank score than the non-contextual word embedding model, FastText [2], although it has higher coverage. Furthermore, the data used for evaluation have no gold standards, and thus the conventional metrics, Precision, Recall and F1, are not computed.

2.3.4 Entity Alignment in Knowledge Graphs

A knowledge graph (KG) typically relies on an ontology as a schema (metadata) that defines the semantics of the dataset expressed by KG. It can be deemed as a collection of entities (nodes) and relationships (labeled edges) between them. The entity alignment or matching (EM) task of KGs often refers to identifying entity equivalence between different KGs because a KG alone has no rich semantics to infer complicated relationships as in OM. However, both KGs and ontologies are graph-like objects, implying that the graph embedding-based approaches used in EM of KGs are likely to be applied (with adaption) to OM as well.

The traditional EM approaches rely on the completeness of ontology semantics because they either utilize equivalence reasoning or similarity computation of symbolic features (e.g., string similarity between entity labels) [30]. Therefore, in the scenario where the to-be-aligned KGs are accompanied by their respective ontologies, traditional OM systems like LogMap and AML can be (almost) directly applied to the task.

The embedding-based EM models adopt *graph embeddings* and/or *word embeddings* to represent entities and the relationships between them. The translational models (e.g., TransE [3]) are conventional approaches for learning graph embeddings with an objective function aiming to minimize some distance metric (e.g., Euclidean distance in $\mathbb{R}^2$) between equivalent entities. More recently, graph neural networks (GNNs) are proposed to learn entity embeddings to incorporate more features such as multi-model data and taxonomies, which have been proven beneficial [4]; word embedding models such as Word2Vec [21] and FastText [2] are also used for encoding the literal information. Nevertheless, compared to EM of KGs, OM is often more abundant in different types of knowledge (textual, structural, and logical), and thus

we can derive more comprehensive entity embeddings that incorporate all sorts of information.

Moreover, we observe that pre-trained language model has also been applied to EM of KGs in a very recent publication that proposes a model named DITTO [17]. To the best of our knowledge, DITTO is the first system that applies BERT to EM of KGs with promising results; and our work, BERTMap, is the first successful OM counterpart. These certainly shed light for future research of entity alignment in both KGs and ontologies with pre-trained language modelling.

2.3.5 Ontology Blocking

Ontology blocking refers to the technique of dividing an ontology alignment task into smaller sub-tasks of reduced complexity, while keeping a high coverage of ground truth mappings. The total number of classes involved in these sub-tasks is likely to be much smaller than the number in the original task because the matched classes are often the minority. For example, the average number of classes in the OAEI LargeBio ontologies (FMA, NCI, and SNOMED) is more than $100K$, while the number of reference mappings is on average $\approx 10K$. To compare more systems (especially the machine learning-based approaches) with less worry of time consumption, dividing an OM task using the ontology blocking (or portioning) technique is needed.

In [12], the authors proposed a blocking approach based on lexical index and semantic embeddings. The lexical index can be seen as a dictionary that has word tokens as *entries* (or *keys*) and classes that contain the corresponding tokens in their labels are the *values*. Classes that can be accessed from the same entry are likely to be matched because their names or aliases have some similarities. The semantic embedding is computed for each key-value pair of the lexical index by concatenating the word embeddings of the tokens in the key and the value, respectively. The semantically closed entries are thus merged into a cluster. In this way, the input ontologies are trimmed to classes involved in the semantic clusters and render a reduced OM task. Results of this work demonstrate that a LargeBio task can be reduced as low as 5% of the original size using the proposed method, while the coverage of mappings is larger than 90%.

Nevertheless, none of the existing ontology blocking approaches take the word-level contexts into consideration. For instance, the lexical index-based technique mentioned in the previous paragraph assumes that all the aligned classes share some word tokens in their labels, but the word meanings vary from its contexts. Similar to what have been employed in BERTMap, we suggest two possible improvements here:

(i) Using sub-word tokens as the entries rather than the full word tokens;

(ii) Adopting contextual token embeddings in the semantic embeddings rather than the non-contextual ones.

The first suggestion will lead to fewer keys but more classes in the values because we ease the restriction from shared word tokens to shared sub-word tokens; the second suggestion will yield semantic clusters that consider word-level contexts. As a result, we can hopefully cover more mappings in the reduced OM task.

Chapter 3

BERTMap

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. Moreover, applying machine learning to OM confronts a fundamental challenge: the number of correct mappings is typically several orders smaller than the wrong mappings. Such extreme imbalance between positive and negative samples makes the supervised learning scheme an impractical choice. So instead of directly learning from training mappings, we should consider an unsupervised or semi-supervised learning scheme with reasonable heuristics such that we can sufficiently exploit the information of ontologies. In this report, we propose BERTMap, a novel ontology alignment system that uses the well-known contextual embedding or language representation model, BERT, to learn text semantics implied by ontologies to predict mappings, and further refine the outputs using the graphical and logical information of ontologies. The workflow of BERTMap is shown in Figure 1.1. In the following sections of this chapter, we describe the technical details of each step.

3.1 Corpus Construction and BERT Fine-tuning

3.1.1 Ontology Corpora

In real-world ontologies, a named class often has multiple labels defined by annotation properties such as *rdfs:label*. They act as the class's aliases. For convenience, we denote a label after preprocessing[1] 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

[1]This includes lowercasing and underscore symbol removing.

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 mappings are given (e.g., annotated by human experts) 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. Hard non-synonyms are not available here because mappings do not involve disjointness. Also, Identity synonyms here because they have been included in the intra-ontology corpus.

Complementary corpus. Besides the input ontologies, 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 corpus, cross-ontology corpus and complementary corpus are denoted as *io*, *co* and *cp*, respectively. Note that *io* is essential, 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 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 corresponding non-synonyms.

3.1.2 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. The inputs of BERT are the tokenized label pairs with the 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 [20]. The final output is of the form $\langle 1 - y, y \rangle$, where $y \in [0, 1]$ is the score that indicates the degree that the input label pairs are synonymous.

3.2 Mapping Prediction

To compute a matched class for each class $c \in C$, a naive solution is to search for $\arg\max_{c' \in C'} P(c \equiv c')$ by traversal. 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 using a mixed strategy that combines the string-match module and the fine-tuned BERT classifier.

3.2.1 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 [13, 32]. In contrast, BERTMap exploits a sub-word inverted index which has the following advantages:

(i) it captures various word forms without extra processing. For instance, the words "tokenization" and "tokenizing" will be transformed to "token" and the corresponding suffixes, namely "##ization" and "##izing".

(ii) it parses unknown words into consecutive known sub-words instead of simply treating them as one special token. For example, suppose "H1N1" (a flu name) is a new word, sub-word tokenization will transform it to the sequence: "H", "##1", "##N", "##1", whereas the traditional word-level method will treat it the same as other unknown words using the same symbol, `<unk>`.

We build sub-word inverted indices based on BERT's inherent WordPiece tokenizer [34], 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 [7], and in this context we consider generality to be preferable to task specificity.

We construct[2] 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 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.

3.2.2 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:

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

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

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 $c' \in C'$;

(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.

3.3 Mapping Refinement

With the predicted mappings and their scores, BERTMap further extends and repairs the mappings by utilizing the graphical and logical information.

3.3.1 Mapping Extension

Recall from Section 2.1 where we state the locality principle that if a source class c and a target class c' are matched, their semantically related classes (such as parents and children) are likely to be matched. BERTMap utilizes this principle to discover new mappings from existing highly scored mappings with an *iterative mapping extension* algorithm (see Algorithm 1). The algorithm starts by initializing a *frontier* set with the high confidence mappings that have been predicted in the previous step. Then, for each mapping (c, c') in the frontier, it computes the Cartesian Products for their parents (super-classes) and children (sub-classes), respectively (Line 8). For each potential matched class pair in these two products, a new mapping is preserved (Line 10-12) only when:

(i) the mapping score $\geq$ the extension threshold κ;

(ii) the mapping is not previously seen (in $\mathcal{M}$ and $\mathcal{M}_{ex}$).

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 Sup$(\cdot)$ be the function that returns superclasses
4: Let 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** $\mathcal{M}_{ex}$

At each iteration, the newly extended mappings will serve as the frontier for the next iteration, until there is no more mappings to be discovered (Line 15-16). Overall, the algorithm iteratively explores 1-hob neighbours that have subsumption relationships with the matched classes, trying to find mappings that have been missed during mapping prediction.

Note that 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 $\kappa = 0.9$.

3.3.2 Mapping Repair

Recall from Section 2.1 that mapping repair aims to remove mappings that will cause 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 (see Section 2.1). However, computing a diagnosis is usually time-consuming, and there may be no unique solution. To address this, [15] proposes a propositional logic-based repair method that can efficiently compute an *approximate* repair $\mathcal{R}$ which ensures that:

(i) $\mathcal{R}$ 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 many classsic OM systems, but is rarely considered in machine learning-based approaches. In this work, we adopt the repair tool developed by [15] as the final step of our mapping refinement process.

Chapter 4

Evaluation

We evaluate BERTMap on three biomedical ontology alignment tasks based on the OAEI LargeBio Track[1]. Internally, we conduct thorough ablation study by comparing BERTMap under *unsupervised* and *semi-supervised* settings, with or without identity synonyms, and with or without the complementary corpus. Moreover, we find that BERTMap is robust to hyperparameter selection. Externally, we compare BERTMap with various baseline systems including both rule-based and machine learning-based models. In particular, we find that BERTMap can often outperform the state-of-the-art ontology alignment systems, LogMap [13] and AML [8], on these datasets.

4.1 Experiment Settings

In this section, we illustrate, in details, the ontology alignment datasets that we use to evaluate BERTMap against other baseline systems. We also present different settings of BERTMap, the evaluation metrics, and the baselines and why we choose them.

4.1.1 Datasets and Tasks

The evaluation first considers the FMA-SNOMED and FMA-NCI small fragment tasks from the OAEI LargeBio Track. Table 4.1 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 the LargeBio Track. We also consider an extended task of FMA-SNOMED, denoted as FMA-SNOMED+, where the target ontology is extended

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

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

by introducing the labels from the latest version of SNOMED.[2] 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 the class set of SNOMED, we extract its labels $\Omega(c)$ and for each label $\omega \in \Omega(c)$, we search for classes in the original SNOMED that have ω as an alias[3]; we then add all the labels of the searched classes back to the LargeBio SNOMED to obtain SNOMED+.

Note that 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 (to enhance the fine-tuning corpora) on the FMA-SNOMED task but for both fine-tuning and prediction on the FMA-SNOMED+ task. This means that on the FMA-SNOMED+ task, not only the BERTMap systems but also the baseline models can be benefited from the complementary labels because they are added back to the input ontology.

4.1.2 Evaluation Metrics

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

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

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

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

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

[3]In SNOMED, aliases are defined through the annotational properties `rdfs:label`, `rdf:about="http://www.w3.org/2004/02/skos/core#altLabel"`, and `rdf:about="http://www.w3.org/2004/02/skos/core#prefLabel"`.

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.

4.1.3 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. To be more specific, 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 [1]. The BERT model is then 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 subsequent mapping prediction. The cut-off of sub-word inverted index-based candidate selection is set to 200. Our implementation uses

(i) `owlready2`[4] for ontology processing;

[4] `https://owlready2.readthedocs.io/en/latest/`.

(ii) `transformers`[5] [33] for BERT; and

(iii) a single `GTX 1080Ti` GPU for training and making prediction.

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 (with the mapping extension threshold κ set to 0.9) 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 $\geq \lambda$.

4.1.4 Baselines

We compare BERTMap with various baselines including rule-based an machine learning-based ones. As mentioned in 3.2.2, the mapping score function of BERTMap applies string-matching on "easy" mappings (i.e., to detect classes that share at least one label), so it is reasonable to include the *(i)* string-matching as an independent baseline model. Mathematically, it computes the mapping score as:

$$S_{\text{string-match}}(c, c') = \begin{cases} 1.0 & \text{if } \Omega(c) \bigcap \Omega(c') \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Moreover, we consider its improved version, the *(ii)* edit-similarity model, as another baseline, which has the following expression:

$$S_{\text{edit-similarity}}(c, c') = \max_{(l,l') \in \Omega(c) \times \Omega(c')} \mathbf{nes}(l, l')$$

where $\mathbf{nes}(\cdot, \cdot)$ refers to the normalized edit similarity (or 1 - (the normalized edit distance[6]) between two strings. Note that this model assigns the *maximum* normalized edit similarity score between label pairs of two classes as their mapping score, and it incorporates all the string-matching cases (i.e., when $\exists\ (l, l') \in \Omega(c) \times \Omega(c')\ s.t.\ \mathbf{nes}(l, l') = 1.0$).

[5] `https://huggingface.co/transformers/`.

[6] Also known as normalized Levenshtein Distance whose information is available at: `https://en.wikipedia.org/wiki/Levenshtein_distance`.

We also consider two pre-trained BERT embedding models as introduced in [23]: the *(iii)* `[CLS]` token embedding model and the *(iv)* mean token embedding model. Recall from the Section 2.2 that BERT's outputs at layer l can be expressed as:

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

The `[CLS]` token embedding model takes $\mathbf{v}_{CLS}^{(L)}$ as the final representation of the whole input sequence whereas the mean token embedding model takes $\overline{f_{bert}}(\mathbf{x}, L)$, i.e., the centroid of all the token embeddings. Note that L is the index of the last layer. We encode each alias sentence of a class using these two strategies, and then average all the resulting sentence embeddings as the class embedding. The mapping score between two classes is thus obtained by computing the *cosine similarity* between their embeddings. The reason for including these two models is to examine if the background knowledge of BERT can be directly transferred to the ontology alignment task. However, our results show that such models perform even worse than the string-matching baseline. Therefore, we opt to not discuss the corresponding results in the result tables (see Section 4.2).

It is important to include the state-of-the-art systems as baselines — in this work, we consider *(v)* LogMap and *(vi)* AML, which are leading systems in many OAEI tracks and other tasks. Furthermore, we consider some variants of LogMap including *(vii)* LogMapLt, the lexical matching part of LogMap; and *(viii)* LogMap-ML*, the machine learning extension of LogMap. Note that LogMap-ML* is a variant of LogMap-ML [6] 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.

The string similarity-based and BERT embedding-based models are our internal baselines, for which we set up the same candidate selection and hyperparameter search procedure as for BERTMap; whereas AML and LogMap-related models are external systems, and we use their default implementation to produce the results.

4.2 Results

The results together with the corresponding hyperparameter settings are presented in Tables 4.2, 4.3 and 4.4, where 90% Test Mappings and 70% Test Mappings refer to the results measured on $\mathcal{M}_{test}$ of the unsupervised setting and $\mathcal{M}_{test}$ of the

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 4.2: 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 4.3: Results of BERTMap under different settings and baselines on the FMA-SNOMED+ task.

semi-supervised setting, respectively. 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

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

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

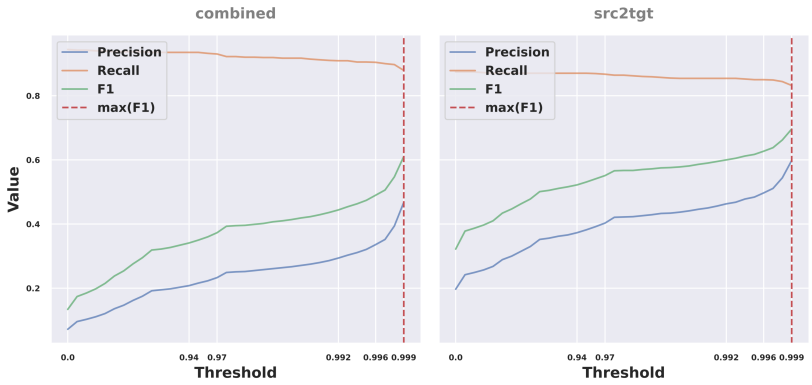

Figure 4.1: Validation results of BERTMap ($io + co + ids$) on the FMA-SNOMED+ task with mapping score threshold λ ranging from 0 to 1. The left and right plots correspond to $\tau =$ `combined` and $\tau =$ `src2tgt`, respectively.

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 hyperparameter selection; most of its settings lead to the same best hyperparameters (i.e. $\tau =$ `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 4.1, 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 (see Section 4.3 for full ablation results).

In Table 4.5, 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.

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

4.3 Mapping Thresholds on Validation Set

In Figure 4.2, 4.3 and 4.4, we present, for all the BERTMap models in this paper, the plots of evaluation metrics (Precision, Recall and Macro-F1) against the mapping threshold $\lambda \in [0, 1)$ on the validation set. Figure 4.3 correspond to (left-to-right, top-to-bottom) the `combined`, `src2tgt`, and `tgt2src` results of io, $io + ids$, $io + co$, $io + co + ids$, $io + ids + cp$, $io + co + ids + cp$ settings on the FMA-SNOMED task. Figure 4.2 and 4.3 correspond to the `combined`, `src2tgt`, and `tgt2src` results of io, $io + ids$, $io + co$, $io + co + ids$ settings on the FMA-NCI task and FMA-SNOMED+ task, respectively.

Note that the validation results are generally worse than testing results because when evaluating on smaller mapping set, we need to ignore more positive mappings whereas the number of negative mappings stays the same, resulting in the prominent drop of F1 score.

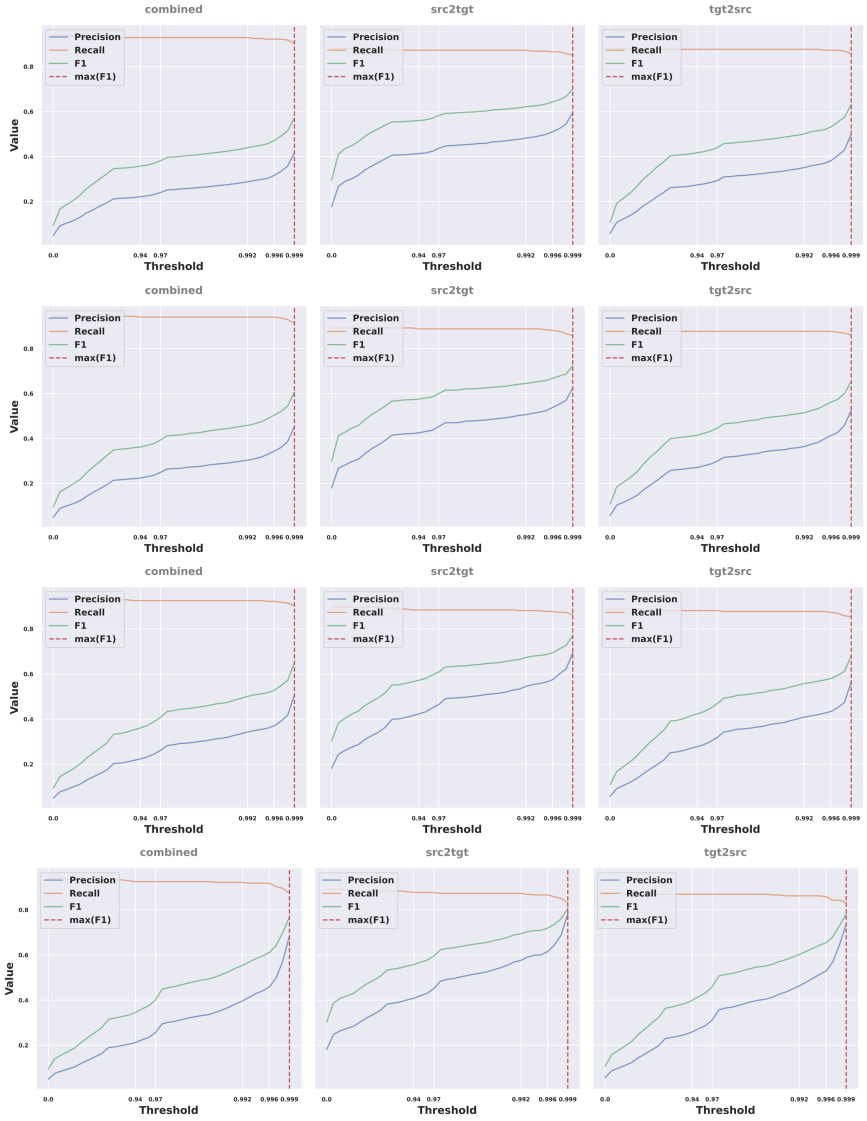

Figure 4.2: Precision, Recall and Macro-F1 of BERTMap on the validation sets of the FMA-NCI task as the mapping score threshold ranges from 0 to 1 (excluded 1 because it represents the sting-match result). The maximum F1 is indicated by a red vertical line.

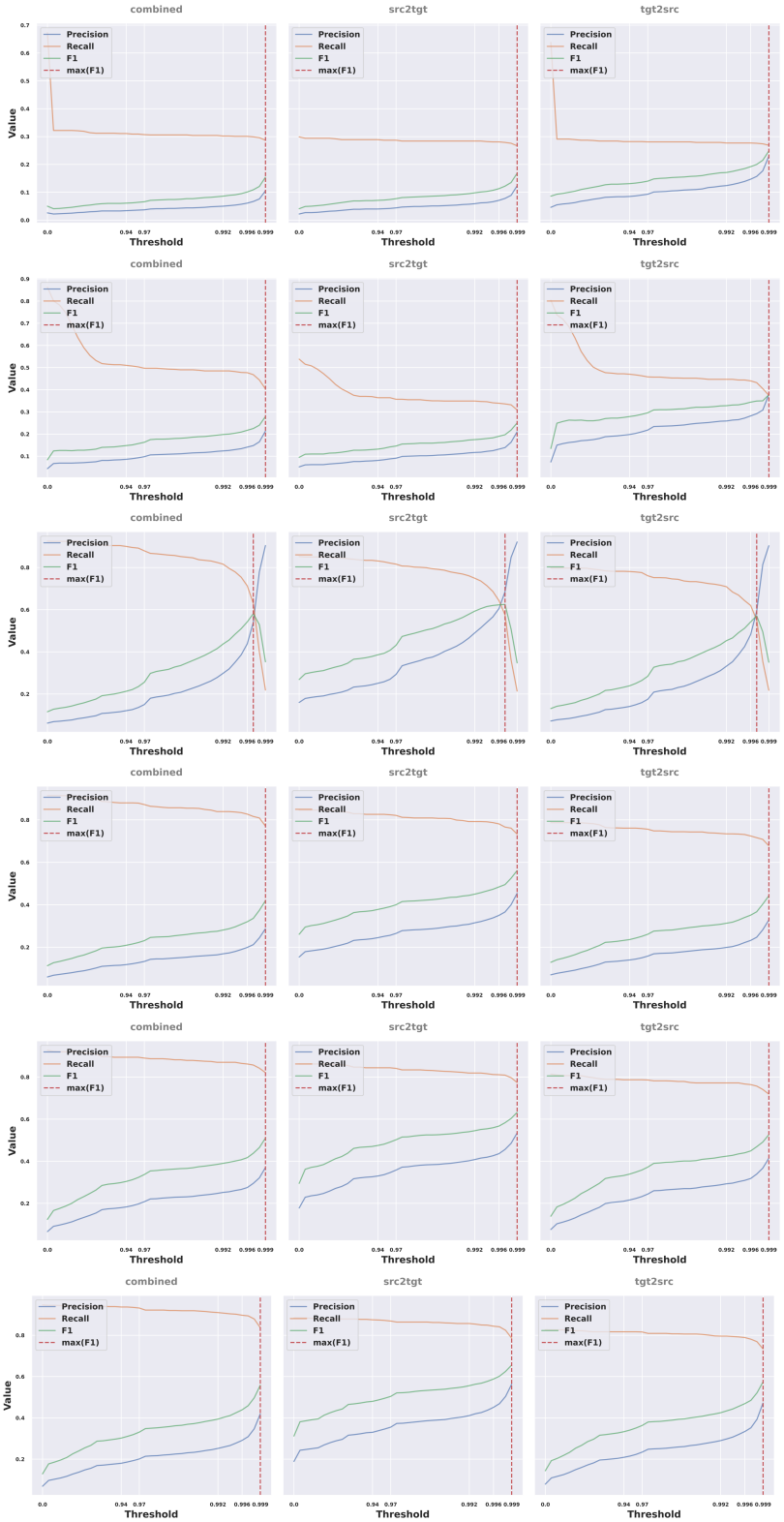

Figure 4.3: Precision, Recall and Macro-F1 of BERTMap on the validation sets of the FMA-SNOMED task as the mapping score threshold ranges from 0 to 1 (excluded 1 because it represents the sting-match result). The maximum F1 is indicated by a red vertical line.

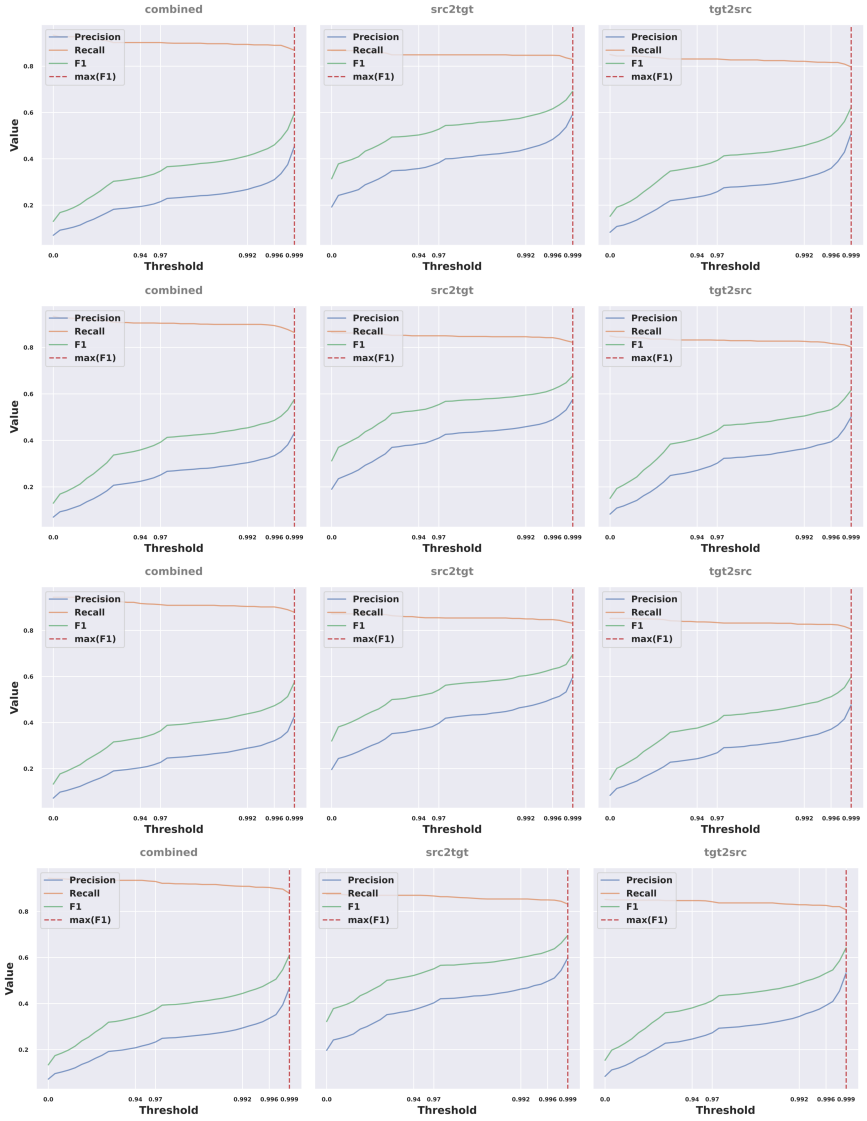

Figure 4.4: Precision, Recall and Macro-F1 of BERTMap on the validation sets of the FMA-SNOMED+ task as the mapping score threshold ranges from 0 to 1 (excluded 1 because it represents the sting-match result). The maximum F1 is indicated by a red vertical line.

Chapter 5

Conclusions & Discussion

In this Chapter, we first reiterate some principles significant to OM system design (Section 5.1); we then discuss if mapping repair is necessary to OM (Section 5.2); finally, we conclude our work with a qualitative analysis of BERTMap from various aspects.

5.1 Principles of OM Systems

An OM system usually needs a classifier or a predictor that can decide whether or not two concepts are semantically related. But in the actual scenario, the classifier or predictor alone is not enough because the potential candidates is typically of a large number of magnitude — this poses two fundamental challenges:

(i) to distinguish concepts that are seemingly close but are actually different is more difficult that decide whether two concepts have some extent of similarity;

(ii) to construct an efficient and effective searching algorithm without sacrificing much recall.

A full-fledged OM system needs to detect concepts that are semantically related out of tens of thousands of options and to be evaluated correspondingly. Evaluation metrics that can sufficiently reveal an OM model's ability in real-world application are Precision, Recall and Macro-F1 **on the mappings**, or their semantic variants that also take logical conflicts into consideration [11].

Current machine learning-based approaches often lack transferability and convincing evaluation scheme. That is, the authors often claimed promising results on some particular tasks using incomplete evaluation methods. For instance, in [18], the authors focus on developing a system that can predict `IS-A` (subsumption) relationships

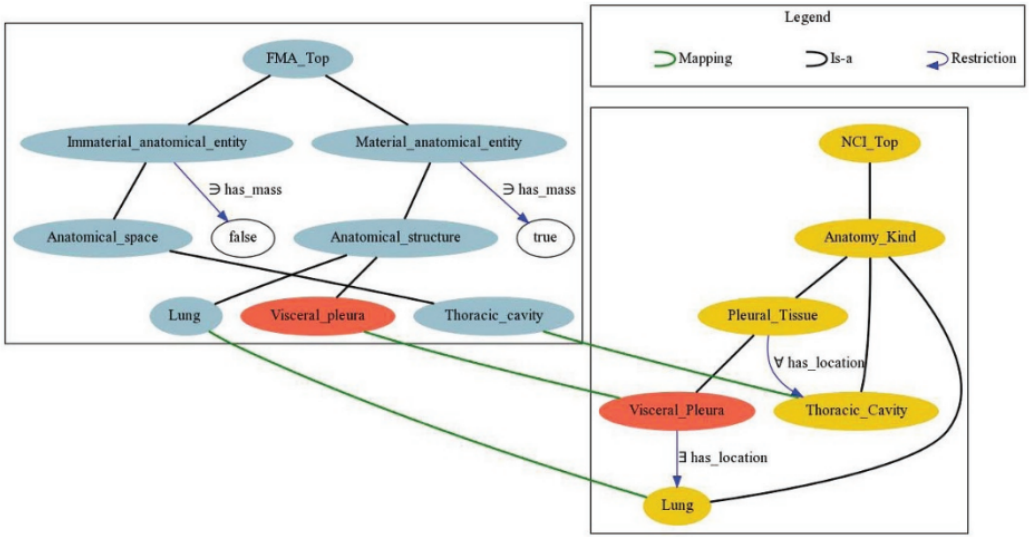

Figure 5.1: Logical inconsistency example after integrating FMA and NCI ontologies through three mappings (indicated by green lines) between the classes named "Visceral Pleura", "Lung" and "Thoracic Cavity" in both ontologies [14].

between concepts of an ontology, such that it can place a new concept into its right position of an existing ontology to achieve ontology maintenance. However, instead of examining the real performance on concept placement, the evaluation is simplified to a classification problem with positive-negative sample ratio set to $1:1$. Achieving prominent scores on such a simplified task is far from the real-world application and thus is not convincing.

In short, when we design a machine learning-based OM system, directly (e.g., supervised learning) or indirectly (e.g., unsupervised or semi-supervised learning), we need to tackle with the challenges mentioned above and evaluate the system fairly. Inspired by the relative "chaos" in evaluating OM systems, especially the machine learning-based ones, we also consider developing an OM evaluation platform that can evaluate and compare different OM systems in various scenarios as one of the future works (see details in Section 6.2).

5.2 Is Mapping Repair Necessary?

Consider an example taken from [14] (see Figure 5.1), there are clear correspondences between the entities "Lung", "Visceral Pleura" and "Thoracic Cavity" in both FMA and NCI ontologies. However, integrating FMA and NCI through these three mappings makes "Visceral Pleura" unsatisfiable (marked by red color). According to NCI,

"Visceral Pleura" is a "Pleural Tissue" located in "Thoracic Cavity", which according to FMA, is a "Immaterial Anatomical Entity". But "Visceral Pleura" is also located in "Lung", which is a "Material Anatomical Entity" specified in FMA. This leads to a contradiction because "Visceral Pleura" is located in an anatomical entity that is both immaterial and material. Nevertheless, according to the knowledge of a biologist, such logical error is not an actual conceptual error but instead a design choice in categorizing biological entities. In this example, "Thoracic Cavity" can be deemed as an immaterial anatomical entity or as an indivisible part of a material anatomical entity depending on how you think of it. Hence, a natural question is how should we justify mapping repair in such cases — is it necessary?

Admittedly, by employing reasoning tools, we can detect potential logical conflicts which may indicate fundamental errors in the original design. We can thus apply mapping repair to achieve quality assurance, which is an essential part of ontology maintenance. However, the inconsistency might happen because of design choices rather than a genuine mistake; and in this case, we need human experts to be involved. The OAEI LargeBio Track adopts the semantic Precision, Recall and F1 which regard inconsistent mappings as neither positive nor negative. Such evaluation actually underestimates the system performance because not all logically incoherent mappings are indeed wrong or should be neglected. In future work, we also need to evaluate on plain Precision, Recall and F1 to obtain an overestimation of the system performance. By combining these two evaluation results, we can possibly have a more comprehensive analysis of the true performance.

5.3 Qualitative Analysis of BERTMap

In this report, we propose a novel ontology alignment system, BERTMap, which exploits the textual, structural and logical information of ontologies. The backbone of BERTMap is its predictor, which utilizes the contextual embedding language representation model, BERT, to learn word semantics and contexts effectively, and computes mapping scores with the aid of sub-word inverted indices. The mapping extension module recalls more mappings, and the repair module removes some mappings that cause logical conflicts for higher precision.

BERTMap suits real-world ontology alignment tasks since it works well under the unsupervised learning setting and can be further improved by given mappings (e.g., from human experts or user interaction) and/or complementary sources. BERTMap has already achieved promising results on three biomedical ontology alignment tasks,

and even outperform the leading OM systems, LogMap and AML. In the following paragraphs, we provide qualitative analysis of BERTMap from various perspectives including *novelty*, *generalizability*, *scalability*, and *evaluation*.

Novelty. BERTMap demonstrates a promising way of utilizing the contextual embedding model in OM while direct applications (i.e., to use pre-trained BERT embeddings or formulate the task in a supervised learning way with SentenceBERT architecture) has been proven not effective in the preliminary work [23]. Moreover, BERTMap requires no sophisticated feature engineering or auxiliary data from various sources, unlike other machine learning-based OM systems, especially those concerning supervised learning. It sufficiently utilizes what the input ontologies can provide (unsupervised), and can be enhanced further by known mappings (semi-supervised) and/or complementary ontology. This suggests that BERTMap has a highly flexible system pipeline with no strict requirement of annotated datasets. Finally, BERTMap considers both text-level contexts (by using the contextual embeddings) and ontology-level contexts (in the graph-based mapping extension algorithm), while to the best of our knowledge, there are no leading OM solutions that consider both of these contexts.

Generalizability & Transferability. BERTMap does not require any domain-specific knowledge and choosing a suitable pre-trained BERT is simply an engineering choice with no major influence on the final performance — what is really important is how we fine-tune the pre-trained BERT and how we use the classifier to make mapping predictions. Since there is no part in the system pipeline that is customized to ontologies of specific domains, BERTMap can be well suited to tasks beyond biomedical domains.

Scalability & Time Consumption. The real time-consuming part is BERT pre-training, which has no cost at all because there are many publicly available pre-trained variants. In contrast, fine-tuning only takes a few epochs (typically 2-4 epochs) which does not take much time even training on outdated GPUs (average around 40 minutes on `GTX 1080Ti` on our datasets). Also, in real practice, OM is an offline process that does not require instant responses — spending this amount of time to obtain the matching results should not be viewed as a major concern. In fact, we actually develop the sub-word inverted index-based candidate selection algorithm to reduce the time complexity of searching to linear.

Evaluation. Regarding the machine learning-based baselines, BERTMap is mainly unsupervised suggesting that it can be evaluated on datasets that have no hold-out

annotated data for training, and thus is not comparable to many machine learning-based OM systems because many of them are supervised. Another concern is that many machine learning-based OM systems have sophisticated engineering choices such that they cannot be easily reproduced without knowing details. On the other hand, some may argue that comparing to LogMap and AML is not representative enough, but these two systems have been well-acknowledged as state-of-the-art OM systems in general, and they have participated in many OM tasks including the OAEI tracks for years, with continuous adaptation and improvement. So our results actually imply several indirect comparisons to other systems that have already been reported in the OAEI LargeBio results. Lastly, choosing the LargeBio dataset is because it is the most challenging task in the OAEI tracks, which already involves ontologies of more than 10K concepts. We reiterate that BERTMap does not involve any essential customization on particular datasets, so if it can work on LargeBio, it can be applied to other datasets in a similar way.

Chapter 6

Future Research Plan

BERTMap is not just a standalone OM system, it actually demonstrates the potential of applying the pre-trained language model to OM, and thus implies a series of research work around this topic. As a result, we are working towards **high-quality data integration with OM-PLM** (Ontology Matching with Pre-trained Language Modelling).

In this chapter, we will illustrate how to extend our work along this direction from several aspects including:

(i) Short-term investigation of what BERTMap have missed in its system design and how to improve them with more comprehensive engineering choices;

(ii) Long-term exploration of a more complicated adaption of the pre-trained language model in the OM system's design;

(iii) Task extension to OM tasks beyond matching class equivalence;

(iv) Ontology evaluation platform that provides fair comparison for both machine learning-based and non-machine learning-based OM systems under various scenarios/settings.

In summary, we intend to construct a general OM model that utilizes the enormous background knowledge provided by the pre-trained language model to address different OM tasks such as equivalence, subsumption, or more complicated relationships between classes, instances, and relations. To better reinforce all the claims and statements around our models, we aim to establish a benchmark work that evaluates several representative OM models in a rigid, reasonable, and fair environment and ultimately provides a publicly acknowledged platform open to all OM systems.

6.1 BERTMap Extension

In this section, we provide some technical aspects (short-term and long-term) of how to extend BERTMap to be a better OM-PLM model. We reiterate some highlights of the current BERTMap as follows:

(i) it is primarily unsupervised and thus can be free from the shortage of annotated data;

(ii) it can be flexibly extended to incorporate annotated data and/or external resources;

(iii) it adopts an indirect way to compare class similarity by constructing a fine-tuning task to learn synonyms and non-synonyms;

(iv) it includes mapping extension and repair whose algorithms can be well-adapted to BERTMap's scoring method but not restricted to it.

For a short-term investigation, we do not intend to reconstruct the system pipeline from scratch but instead search for improvement while keeping the above characteristics; for a long-term exploration, we wish to keep *(i)* and *(ii)* while integrating *(iii)* and *(iv)* to build a more compact model with class representations that encode textual, structural, and logical information altogether.

6.1.1 Transitive Property

Recall that we have considered *reflexive* (identity synonyms) and *symmetrical* properties of synonyms and non-synonyms, but we have not considered the *transitive* property, i.e., if ω and ω' are synonymous (resp. non-synonymous), ω' and ω'' are synonymous (resp. non-synonymous), then ω and ω'' should also be regarded as a synonym (resp. non-synonym) pair. However, considering such property will drastically increase the size of the training corpora, so we need a more careful strategy to adopt this idea in our future work.

6.1.2 Negative Sampling

Negative sampling is essential to a classification task because high-quality negative samples can save considerable training time while preserving performance. In BERTMap, we adopt a relatively naive solution by categorizing non-synonyms into two types with soft non-synonyms extracted from random pairs of classes and hard

non-synonyms extracted from sibling classes. But in fact, we could develop a more sophisticated negative sampling approach from, e.g., establishing a more comprehensive disjointness heuristic and emphasizing more on the semantically ambiguous pairs. For extremely large-scale ontologies, we also need some sampling technique on the positive samples.

6.1.3 Class Subsumption

As indicated in Section 2.1, BERTMap aims to solve cross-ontology class equivalence, but ontology alignment, in general, is not restricted to this relationship. An ongoing research project of ours is BERTSubsumption which adopts the similar techniques as in BERTMap to address cross-ontology subsumption. The major differences are:

(i) Instead of training on synonymous or non-synonymous label pairs, we are now shifting to label pairs from parent-children classes.

(ii) Hard non-synonyms can be extracted from uncle-nephew classes.

(iii) Symmetrical property is invalid here unlike in the equivalence relationship.

(iv) We could possibly examine if BERT has the ability of probabilistic reasoning because in the case when class c is subsumed by class c' and vice versa, can we determine if c is equivalent to c'?

Moreover, instead of training on isolated class pairs, we can formulate the BERT inputs as also including the ontology contexts such that it becomes a long-text classification problem.

6.1.4 Prompt Learning

Recall from Chapter 3 that $\Omega(\cdot)$ is the function that returns preprocessed labels of an input class, and ω represents an individual label, we illustrate the current BERTMap prediction in Figure 6.1; the mapping score between class c and c' is given by the average of all the synonym probabilities (predicted by the fine-tuned BERT classifier) between their respective labels. This relatively simple approach has been proven effective in our evaluation results but it is far from exploiting the full potential of BERT. To be specific, it now considers isolated classes with isolated labels, meaning that each prediction is conducted on a single label pair without incorporating ontology-level contexts. In fact, considering isolated labels helps to keep the relative

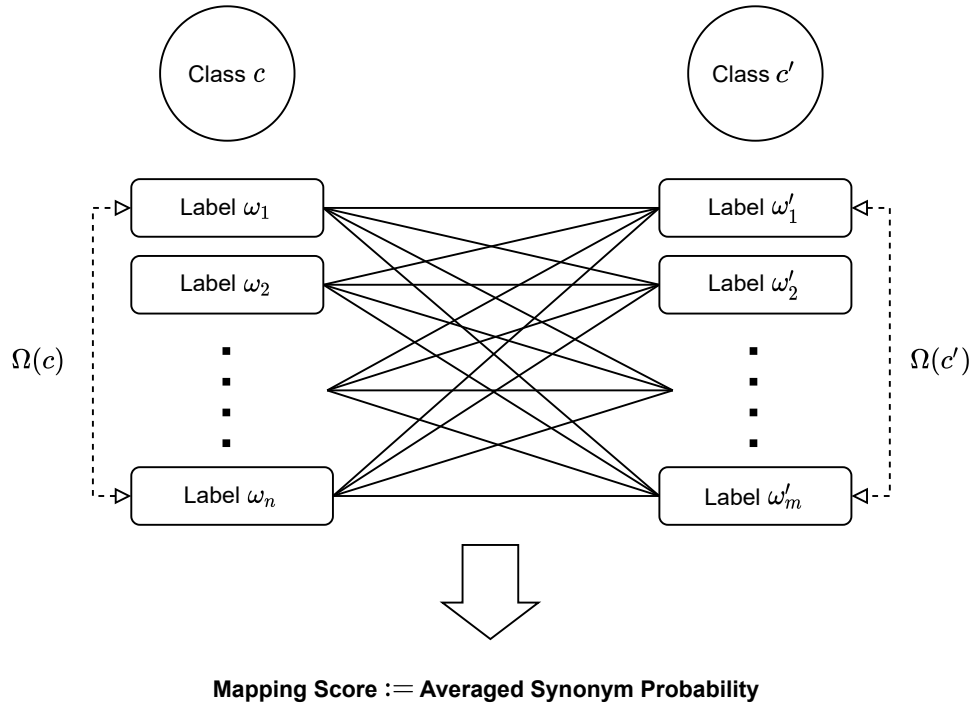

Figure 6.1: Illustration of Mapping Prediction of current BERTMap.

input scale because an ontology does not provide a constant number of labels for each class; but we should consider the ontology-level contexts by e.g., formatting the information of neighbouring classes as a part of BERT's input.

We first present a naive approach using long-text classification. Suppose class c has classes a and b as its neighbours, class c' has classes a' and b' as its neighbours, then we can format the BERT input as follows:

> `[CLS]` $T(\omega_c)$ `[EOS]` $T(\omega_a)$ `[EOS]` $T(\omega_b)$ `[SEP]` $T(\omega_{c'})$ `[EOS]` $T(\omega_{a'})$ `[EOS]` $T(\omega_{b'})$

where the first sentence is given by "$T(\omega_c)$ `[EOS]` $T(\omega_a)$ `[EOS]` $T(\omega_b)$" and the second sentence is given by "$T(\omega_{c'})$ `[EOS]` $T(\omega_{a'})$ `[EOS]` $T(\omega_{b'})$", $T(\cdot)$ returns the sequence of sub-word tokens after tokenzing some label of a class, `[EOS]` is a new special token used for separating tokenized label sequence of different classes. In this manner, to predict if a label pair, ω_c and $\omega_{c'}$, are synonymous, it also needs to consider the correspondences among the neighbours. To investigate if this method actually works, we need to explore:

(i) How do we select meaningful neighbours (e.g., only those with subsumption relationships) and how many of them should we consider?

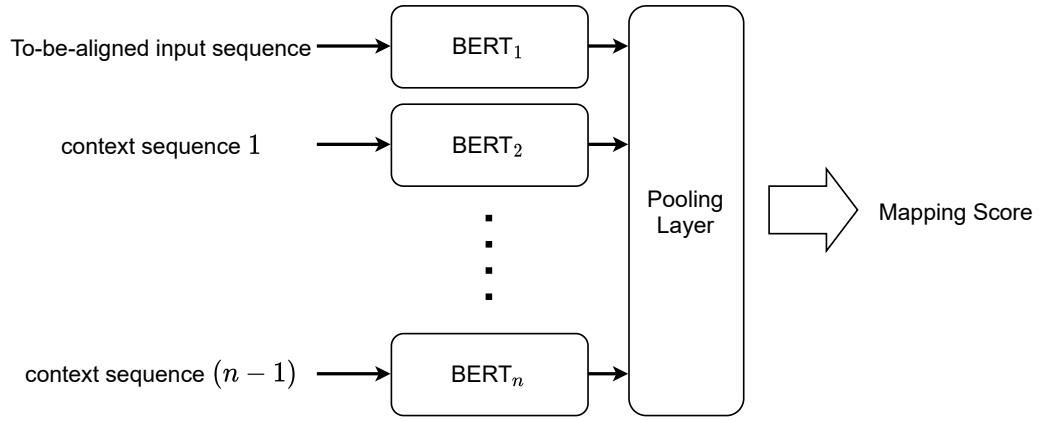

Figure 6.2: Illustration of Prompt Learning Using BERT Ensembling.

(ii) How long should we train for BERT to "embrace" the new special token `[EOS]`?

(iii) If we again consider an isolated label to represent a class in the input of BERT, the number of possible combinations will be drastically increased after considering the neighbours — what sort of sampling technique needs to be employed here?

Such learning scheme is referred to as *prompt learning* because we use the labels of neighbouring classes as "prompts" to help the synonym classification. It is recommended to read a full survey of prompt learning in [19] for various interesting applications. Nevertheless, the "prompting" method introduced above is still relatively simple because we cannot emphasize the labels of to-be-aligned classes — an even further development is to leverage model ensembling.

In Figure 6.2, we propose using BERT ensembling to predict the mapping score between class c and class c', where we use n fine-tuned BERT to encode the input sequence of to-be-aligned classes and the context sequences associated to them. Subsequently, the n hidden representations are merged through a pooling layer to compute the final mapping score. In this way, we can easily attach more importance to the main input sequence by adjusting the pooling weights, and for different BERTs we can consider ontology-level contexts distinguished by e.g., different types of relationships and/or semantic locality.

6.2 OM Evaluation

As discussed in Section 5.1, OM evaluation is not strictly regulated among the machine learning-based systems and there is no public platform available for comparing non-machine learning-based OM models to machine learning-based ones. Thus, another direction of our future work will be focusing on creating comprehensive OM benchmarks with various evaluation settings aimed for different purposes. The most general scenario will be evaluating on Precision, Recall and F1 or their semantic variants that consider logical coherence. We can also evaluate systems on Hit@K and MRR which put more emphasis on providing suggestions of possible alignments rather than detecting the exact matches. However, to compute a meaningful Hit@K, we need a sufficient number of high-quality negative samples that are rather similar to the correct match, e.g., classes from the 1-hob neighbourhood. Nevertheless, many existing OM papers take this for granted and thus the evaluation results are far from being convincing.

For machine learning-based models, we need to split data into training, validation and testing sets with reasonable ratios (e.g., $2:1:7$ as used in evaluating BERTMap under the semi-supervised setting) for fair comparisons.

For OM data, we are interested in UMLS-based ontologies such as those used in OAEI tracks and the Mondo Disease Ontologies[1] because they are annotated by human experts and thus can provide trust-worthy ground truth mappings. We will also consult domain experts to verify the data to further ensure the benchmark quality.

[1] `http://www.obofoundry.org/ontology/mondo.html`

Bibliography

[1] Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, June 2019.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[4] Yixin Cao, Zhiyuan Liu, Chengjiang Li, Zhiyuan Liu, Juan-Zi Li, and Tat-Seng Chua. Multi-channel graph neural network for entity alignment. In *ACL*, 2019.

[5] Jiaoyan Chen, Pan Hu, Ernesto Jiménez-Ruiz, Ole Magnus Holter, Denvar Antonyrajah, and Ian Horrocks. Owl2vec*: Embedding of owl ontologies. *Mach. Learn.*, 110:1813–1845, 2021.

[6] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, Denvar Antonyrajah, Ali Hadian, and Jaehun Lee. Augmenting ontology alignment by semantic embedding and distant supervision. In *European Semantic Web Conference*, pages 392–408. Springer, 2021.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[8] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F. Cruz, and Francisco M. Couto. The agreementmakerlight ontology matching system.

In Robert Meersman, Hervé Panetto, Tharam Dillon, Johann Eder, Zohra Bellahsene, Norbert Ritter, Pieter De Leenheer, and Deijing Dou, editors, *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, pages 527–541, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[9] I. Horrocks, Jiaoyan Chen, and Jaehun Lee. Tool support for ontology design and quality assurance. 2020.

[10] Vivek Iyer, Arvind Agarwal, and Harshit Kumar. Veealign: a supervised deep learning approach to ontology alignment. In *OM@ISWC*, 2020.

[11] Qiu Ji, Zhiqiang Gao, Zhisheng Huang, and Man Zhu. Semantic precision and recall for evaluating incoherent ontology mappings. In Runhe Huang, Ali A. Ghorbani, Gabriella Pasi, Takahira Yamaguchi, Neil Y. Yen, and Beijing Jin, editors, *Active Media Technology*, pages 338–347, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[12] Ernesto Jiménez-Ruiz, Asan Agibetov, Jiaoyan Chen, Matthias Samwald, and Valerie V. Cross. Dividing the ontology alignment task with semantic embeddings and logic-based modules. *ArXiv*, abs/2003.05370, 2020.

[13] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist, editors, *The Semantic Web – ISWC 2011*, pages 273–288, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[14] Ernesto Jiménez-Ruiz, B. C. Grau, I. Horrocks, and Rafael Berlanga Llavori. Logic-based assessment of the compatibility of umls ontology sources. *Journal of Biomedical Semantics*, 2:S2 – S2, 2011.

[15] Ernesto Jiménez-Ruiz, Christian Meilicke, B. C. Grau, and I. Horrocks. Evaluating mapping repair systems with large biomedical ontologies. In *Description Logics*, 2013.

[16] Prodromos Kolyvakis, Alexandros Kalousis, and Dimitris Kiritsis. DeepAlignment: Unsupervised ontology matching with refined word vectors. In *Proceedings of NAACL-HLT*, pages 787–798, June 2018.

[17] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang Chiew Tan. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, 14:50 – 60, 2020.

[18] Hao Liu, Yehoshua Perl, and James Geller. Concept placement using bert trained by transforming and summarizing biomedical ontology structure. *Journal of biomedical informatics*, page 103607, 2020.

[19] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, 2021.

[20] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2017.

[21] Tomas Mikolov, Kai Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

[22] Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California, June 2016. Association for Computational Linguistics.

[23] Sophie Neutel and M. D. Boer. Towards automatic ontology alignment using bert. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, 2021.

[24] Ikechukwu Nkisi-Orji, Nirmalie Wiratunga, Stewart Massie, Kit-Ying Hui, and Rachel Heaven. Ontology alignment based on word embedding and random forest classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 557–572. Springer, 2018.

[25] Lorena Otero-Cerdeira, Francisco J Rodríguez-Martínez, and Alma Gómez-Rodríguez. Ontology matching: A literature review. *Expert Systems with Applications*, 42(2):949–971, 2015.

[26] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *ArXiv*, abs/1908.10084, 2019.

[27] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

[28] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[29] P. Shvaiko and J. Euzenat. Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):158–176, 2013.

[30] Zequn Sun, Qingheng Zhang, Wei Hu, Chengming Wang, Muhao Chen, Farahnaz Akrami, and Chengkai Li. A benchmarking study of embedding-based entity alignment for knowledge graphs. *Proceedings of the VLDB Endowment*, 13:2326 – 2340, 2020.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[32] Lucy Wang, Chandra Bhagavatula, Mark Neumann, Kyle Lo, Chris Wilhelm, and Waleed Ammar. Ontology alignment in the biomedical domain using entity definitions and context. In *Proceedings of the BioNLP 2018 workshop*, pages 47–55, July 2018.

[33] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[34] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan

Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, 2016.

[35] Chuncheng Xiang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. Ersom: A structural ontology matching approach using automatically learned entity representation. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2419–2429, 2015.

[36] Han Xiao. bert-as-service. `https://github.com/hanxiao/bert-as-service`, 2018.