Fine-grained Entity Typing without Knowledge Base[*]

Jing Qian[1], **Yibin Liu**[2], **Lemao Liu**[3], **Yangming Li**[3],
Haiyun Jiang[3], **Haisong Zhang**[3], **Shuming Shi**[3]
[1]University of California, Santa Barbara
[2]Peking University [3]Tencent AI Lab
jing_qian@cs.ucsb.edu liuybinpku@pku.edu.cn
{redmondliu,newmanli,haiyunjiang}@tencent.com
{hansonzhang,shumingshi}@tencent.com

Abstract

Existing work on Fine-grained Entity Typing (FET) typically trains automatic models on the datasets obtained by using Knowledge Bases (KB) as distant supervision. However, the reliance on KB means this training setting can be hampered by the lack of or the incompleteness of the KB. To alleviate this limitation, we propose a novel setting for training FET models: FET without accessing any knowledge base. Under this setting, we propose a two-step framework to train FET models. In the first step, we automatically create pseudo data with fine-grained labels from a large unlabeled dataset. Then a neural network model is trained based on the pseudo data, either in an unsupervised way or using self-training under the weak guidance from a coarse-grained Named Entity Recognition (NER) model. Experimental results show that our method achieves competitive performance with respect to the models trained on the original KB-supervised datasets.

1 Introduction

Entity Typing is a fundamental task in Natural Language Processing. Traditional entity typing research focuses on a limited number of entity types while recent studies strive for finer granularity. A major challenge in fine-grained entity typing (FET) is the absence of human-annotated data. To address this problem, a common practice is to seek distant supervision from knowledge bases. Typically, the training data is obtained by linking entity mentions and drawing their types from knowledge bases. For example, Ling and Weld (2012) utilize the information encoded in anchor links from Wikipedia text along with the type information in Freebase (Bollacker et al., 2008). After that, classification models are trained using the resultant FET dataset, as illustrated in the left part of Figure 1.

[*]The first two authors (Jing and Yibin) contributed equally to this work during the internships at Tencent AI Lab.

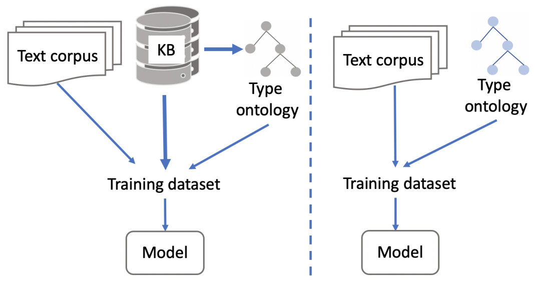

Figure 1: A comparison between the previous FET framework (left) and our proposed FET framework (right). In the previous one, the type ontology is coupled with KB whereas it is not in ours.

Despite its success, this framework has one limitation: The type ontology for training and testing is strongly restricted by the underlying KB. As a result, the application of the trained models under this framework can be hampered by the lack or the incompleteness of the KB (Choi et al., 2018). In real-world applications of FET, the ontology of the fine-grained types depends on the applications instead of the KB. Typically, there may not exist a one-to-one mapping from the ontology of applications to that of the existing KB. For example, a game-related application may need to identify the named entities of some specific video games, such as the heroes and items in DOTA 2. However, there are no knowledge bases covering these named entities. The existing FET framework can not be used when the desired type topology is not compatible with any KB. Worse than this, large-scale knowledge bases are even not available in some languages. As a result, the trained FET classifiers under this framework cannot be effectively applied to real-word applications.

To lift this limitation, we study a novel FET setting, **fine-grained entity typing without a knowledge base**, which is a common setting in real-world applications. Given a fine-grained type ontology defined by an application, we aim to assign a list of labels for each mention within an input sentence, *yet without accessing any knowledge base*. Under this setting, we propose a novel FET framework by leveraging a large unlabeled dataset, as illustrated in the right part of Figure 1. Our proposed framework disentangles the type ontology from the KB and thus the given ontology can be an arbitrary one, defined by a practical application that does not correspond to any KB or by a specific KB as the conventional FET setting. Therefore, our proposed framework is more flexible than the previous one.

Our framework consists of two steps: creating pseudo data from an unlabeled dataset and (iteratively) training the FET model based on the pseudo dataset. In the first step, we aim to create a large pseudo dataset with fine-grained type labels. We propose an automatic method using Hearst patterns (Hearst, 1992) to achieve this goal. There exists a difference between the pseudo data and the testing data in that each sentence in the pseudo data satisfies a (relaxed) Hearst pattern while few testing data do. Consequently, directly training a FET model on the pseudo data may limit its generalization ability. We thereby propose a self-training method with weak supervision from a coarse-grained NER model in the second step to alleviate this issue. We do not focus on designing novel models under the conventional FET setting in this work. Actually, our proposed two-step framework does not set a limit on the backbone FET models and existing FET models can be freely migrated to our framework. Experimental results show that without utilizing any KB in the conventional framework, our proposed framework achieves comparable results to the conventional framework using the same backbone FET model.

Main contributions of our work are:

- We study fine-grained entity typing based on a new setting: FET without a knowledge base.
- Under the new setting, we propose a two-step approach (pseudo data generation and model training) to tackle the FET task.
- Compared with training on KB-supervised datasets, our proposed approach achieves competitive performance on two popular FET testing datasets.
- To facilitate future research under this new setting, we make the created data publicly available. [1]

[1]The dataset will be available at `https://github.com/lemaoliu/fet-data`. Part of this work is implemented in the public system TexSmart at `https://texsmart.qq.com`.

2 Related Work

2.1 Fine-grained Entity Typing

Starting from hand-crafted features (Ling and Weld, 2012; Yosef et al., 2013; Gillick et al., 2014), research on FET has moved to distributed representations (Yogatama et al., 2015; Ren et al., 2016a,b; Zhang et al., 2018) and more advanced neural network models (Dong et al., 2015; Shimaoka et al., 2016a,b; Murty et al., 2018). Shimaoka et al. (2016a) propose the first attentive neural model that outperforms feature-based methods with a simple cross-entropy loss. They use LSTM (Hochreiter and Schmidhuber, 1997) for context encoding. Murty et al. (2018) employ Convolutional Neural Networks to encode the entity mention and context. More recently, pretrained language models, such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), have been employed to achieve better performance (Lin and Ji, 2019; Chen et al., 2020; Onoe et al., 2021). This research line relies on a knowledge base to automatically create training data, since it is challenging to build the human-annotated data (Ling and Weld, 2012; Gillick et al., 2014; Li et al., 2021a)

Our contributions in this work are orthogonal to the above research line. As stated in Section 1, the focus of this work is not to develop more advanced models under the conventional FET framework. Actually, these models can be freely migrated to our proposed framework. It is worth noting that using Hearst patterns for FET was pioneered by Del Corro et al. (2015). However, they employed Hearst patterns as well as a knowledge base to build an FET system. Hence they did not prove that it is feasible to address FET without any knowledge bases as we do.

2.2 Fine-grained Type Ontology

Recent work introduced fine-grained type ontologies based on knowledge bases. Ling and Weld (2012) derive the type ontology consisting of 122 types from Freebase types. Later work (Murty et al., 2017) introduces a even more fine-grained ontology, which consists of over 1,941 types, obtained by manually annotating a mapping from 1,081 Freebase types to WordNet. Rabinovich and Klein (2017) derive the ontology from the Wikipedia categories and WordNet graphs. Del Corro et al. (2015)

use the ontology derived from the entire WordNet hierarchy. Although driving an ontology from the type hierarchy of a knowledge base is an efficient way to build type ontology, it is not practical in real-world application. Instead, the desired types and ontology depend on the specific application and usually there does not exist a one-to-one mapping from the desired types to KB types.

2.3 Free-form Entity Typing

Choi et al. (2018) propose an alternative to the conventional FET framework. Their proposed ultra-fine entity typing task aims to predict free-form types such as noun phrases that describe appropriate types for the role the target entity plays in the sentence. To address this free-form typing task, they automatically create a large dataset using some heuristic methods. In addition, Dai et al. (2021) extend the dataset in Choi et al. (2018) with pre-trained langauge models and then train the typing task on the extended dataset. Since the entity types are replaced by free-form noun phrases, there is no a unified type ontology in their task. However, a formal, explicit specification of the target types plays an important role in the downstream tasks of fine-grained entity typing. Different from their work, our proposed task retains the type ontology as an input for the task, but disentangles it from the knowledge base so as to provide larger flexibility.

3 Problem Statement for FETw/oKB

Suppose $\mathcal{L}$ is a fine-grained entity type ontology for a real-world application as illustrated in Figure 2, which consists of a set of formal types organized as hierarchical trees, and a dictionary used to explain each node in the trees. Each node in a tree is a formal entity type l and each directed edge (l_1, l_2) in the tree indicates that l_2 is a subtype of l_1. Each element in the dictionary consists of a formal type (i.e., a node) and a list of informal type names explaining the meaning of the formal type. Figure 2 shows an example, where the type names of the type "/organization/company" is the list "[company, enterprises, companies, enterprise]" and the type names of "/disease" is the list "[disease, illnesses]".

The generic FET task aims to predict the type set T whose element is in the given ontology $\mathcal{L}$. Note that T may be a set with one or more type labels, so this task is characterized as a multi-class multi-label classification task. Formally, the FET task is modeled by a probabilistic model $P(l \mid x, c)$. A common practice is to parameterize the probabilistic model by a binary classifier $P_\theta(l \mid x, c)$, which indicates whether l is a type label of the mention x. During inference, the classifier outputs all the labels l satisfying the condition $P_\theta(l \mid x, c) > 0.5$.

In the conventional framework, θ is trained on a distantly supervised dataset, which is dependent on a knowledge base. As a result, the trained classifier can only predict the types corresponding to the types in that knowledge base, limiting its application in practice.

Problem Statement In this paper, we focus on FET under a more practical setting, i.e. **Fine-grained Entity Typing without Knowledge Base** (FETw/oKB). That is, given an ontology $\mathcal{L}$, we aim to infer the type set T for the mention x within a sentence c when there are no KBs available.

Relation to Zero-shot FET Suppose the type ontology $\mathcal{L}$ is divided into two disjoint sub-ontology $\mathcal{L}_1$ and $\mathcal{L}_2$, and there is a labeled training dataset where the labels are all from $\mathcal{L}_1$. Zero-shot FET aims to make a prediction on testing mentions whose labels are within $\mathcal{L}_2$. A surge of efforts have been devoted to zero-shot FET (Ma et al., 2016; Xian et al., 2019; Ren et al., 2020; Chen et al., 2021), but all of them assume that training data is obtained from knowledge bases by distant supervision, similar to the conventional FET. Therefore, the proposed FETw/oKB is clearly different from zero-shot FET.

Road Map Generally, it is challenging to train θ under the FETw/oKB setting. In the next sections, we propose a new framework to address FETw/oKB. Its key idea is a two-step approach:

- We propose an automatic method to create pseudo data from a large-scale unlabeled corpus (§4).
- We propose a method to train the classifier by leveraging the pseudo data (§5).

4 Pseudo Data from Scratch

We start from the ontology $\mathcal{L}$ given by a real-world application, and our purpose is to create a large pseudo dataset with fine-grained labels from massive unstructured data. To this end, we propose an automatic method to achieve this goal without manual labor. At a high level, the key idea of our method is illustrated in Figure 2. In the next, we describe the key steps within our method.

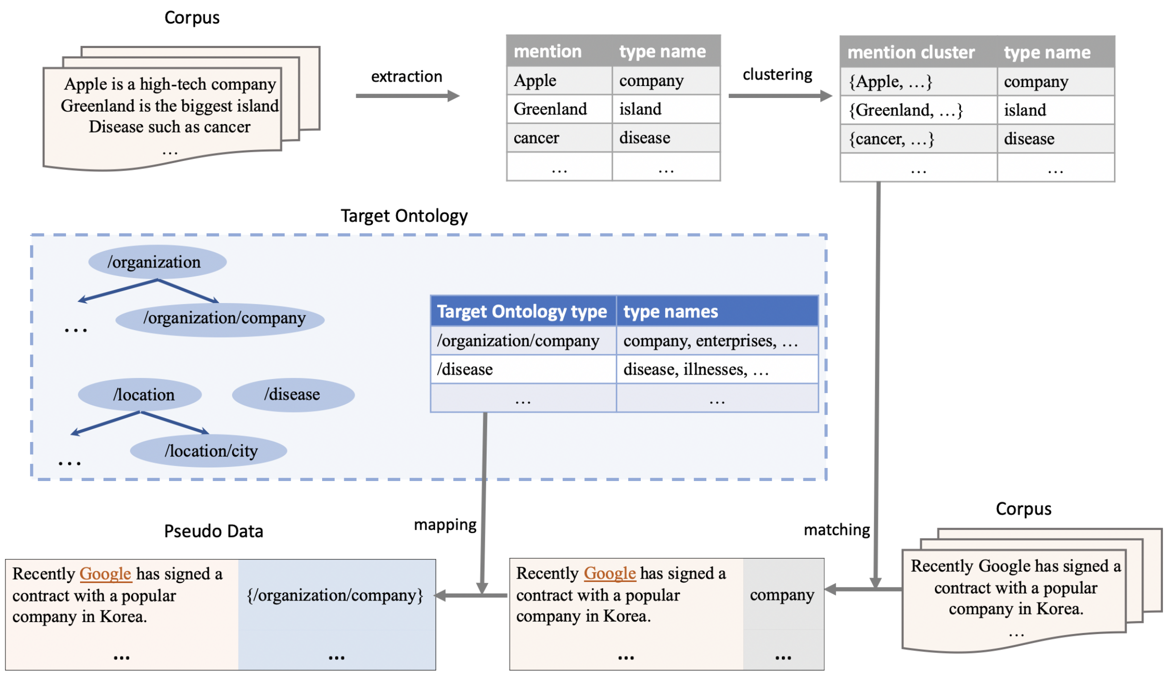
Corpus
Apple is a high-tech company
Greenland is the biggest island
Disease such as cancer
…
extraction
mention | type name
Apple | company
Greenland | island
cancer | disease
clustering
mention cluster | type name
{Apple, …} | company
{Greenland, …} | island
{cancer, …} | disease
Target Ontology
/organization
/organization/company
/location
/disease
/location/city
Target Ontology type | type names
/organization/company | company, enterprises, …
/disease | disease, illnesses, …
mapping
matching
Pseudo Data
Recently Google has signed a contract with a popular company in Korea.
{/organization/company}
company
Corpus
Recently Google has signed a contract with a popular company in Korea.

Figure 2: The process of generating pseudo data. It could be divided into several parts: extracting pairs from the unstructured corpus, clustering the extracted pairs, and matching corpus with clusters to get the pseudo data. See section 4 for a detailed explanation.

4.1 Dictionary Generation

Extraction First, we use Hearst patterns (Hearst, 1992) to generate a mapping dictionary between mentions and type names. Each pattern corresponds to a relationship between a hypernym and a hyponym. For example, the patterns include "NP such as NP" and "NP is a NP". Using these patterns to filter the unstructured data, we collect a large-scale text corpus consisting of sentences such as "An apple is an edible fruit." and "Apple is a high-tech company.". Along with the corpus, we also collect a large number of (mention, type name) pairs such as (apple, fruit), (Apple, company), (banana, fruit), and (Microsoft, company). [2] We denote this set of (mention, type name) as S.

Clustering There might be some pairs in S, which share the same mention but have different type names. Such a mention is an ambiguous mention. For example, "apple" is an ambiguous mention, because both of the type names, "company" and "fruit", can be mapped to it. To resolve this problem, we cluster all mentions in M by an overlapping clustering algorithm and then assign a type name to each mention cluster, whose meaning may be less ambiguous. Standard overlapping clustering algorithms either need to pre-define the number of clusters or are inefficient (Jain et al., 1999). Instead, we design a new algorithm whose basic idea includes three steps: for each mention, it computes its neighbor set consisting of its top k mentions according to a term similarity score; then a hierarchical clustering algorithm (Brown et al., 1992) is applied to each neighbor set, leading to many small subsets in total; then it repeatedly merges a pair of subsets to a larger subset according to a pre-defined threshold and finally take all subsets as clusters. Since one neighbor set may overlap with another one, the resulted clusters can be overlapping. According to the clusters, we create a map from mention clusters to type names. For example, the cluster "[Apple, Microsoft, Google, Facebook, ...]" is mapped to "company". There are unambiguous mentions such as "Google" and "Microsoft" in this cluster, so the ambiguity of "Apple" can be reduced.

It is worth mentioning that Zhang et al. (2020); Liu et al. (2021) adopt a similar method to obtain a mention cluster map for fine-grained entity classification where entity mentions are unknown. In their work, the map is employed to infer the labels by exact matching during inference. Consequently, their work can not address those mentions which are not covered in the map. Instead of directly us-

[2]We implement an in-house extraction algorithm similar to that in NLTK (Loper and Bird, 2002).

ing the map for inference, we employ the map to generate pseudo data for training a typing model, as will be described later. In this way, the model is able to generalize on unseen mentions.

4.2 Pseudo Data Generation by Matching & Mapping

By using the cluster dictionary to match the corpus, we get our pseudo data. Take the sentence *"Recently Google has signed a contract with a popular company in Korea."* as an example. The mention "Google" appears in the cluster [Microsoft, Apple, Google, ...] and this cluster is mapped to the type name "company" by the dictionary. To further preserve high precision in the pseudo data, we perform a calibration strategy when matching: only if we find type name "company" in the context of "Google", the example ([Recently, Google, has signed a contract with a popular company in Korea.], company) will be accepted. In this step, the type name "company" will be used as a calibration to reduce ambiguity. The final step is to map the informal type name "company" to the formal type "/organization/company" in $\mathcal{L}$ according to the ontology dictionary within $\mathcal{L}$. Finally, we obtain pseudo examples such as ([Recently, Google, has signed a contract with a popular company in Korea.], /organization/company).

Due to the calibration strategy, each sentence in the pseudo data has some type name in it, such as "company" in the example above, which is strongly associated with the ground truth type label of the mention, such as "/organization/company" in the example above. Therefore, training models directly on the pseudo data can not result in an effective FET model. Instead, we mask the type name in the pseudo data when using it for training, which leads to about +1 F1 gains in our preliminary experiments.

It is worth mentioning that it is possible to generate pseudo data by only using Hearst patterns along with the ontology dictionary within $\mathcal{L}$, which is much simpler than our proposed generation method. However, this simple generation method leads to limited data compared with ours, because it only extracts those sentences that strictly satisfy a Hearst pattern but ignores some high-quality sentences. Take *"Recently Google has signed a contract with a popular company in Korea."* as an example, our method can match this sentence and identify an entity "Google" with labeled as "/organization/company" even if this sentence does not satisfy any Hearst patterns at all. In this sense, our generation method can be considered as a relaxed version of the Hearst pattern method.

5 Self-Training via Weak Guidance

Note that each sentence in the above pseudo data satisfies a relaxed Hearst pattern while few in the test dataset do. In other words, there exists a gap between the pseudo dataset and the testing dataset.Therefore, directly using the pseudo data for training may not result in a typing model with good generalization ability.

To alleviate this problem, one may try to apply self-training, where the typing model is used to expand the pseudo dataset iteratively during training. However, the typing model can only do part of the job because it is trained to predict the entity type given an entity mention but not to predict the boundary of an entity mention given an input sentence.

We use an off-the-shelf coarse-grained NER model to fill the gap and propose a weakly-guided self-training method. The basic idea is to use the coarse-grained NER model to identify the entity mentions in an input sentence, which are then fed into the fine-grained typing model for self-training. Typically, a coarse-grained NER model can achieve both high precision and high recall on the task of identifying mention boundaries, so the missing mentions in the pseudo data can be found by it. Besides the boundaries of entity mentions, the coarse-grained NER model also predicts a coarse-grained entity type for each mention, which is used as weak supervision in our method to improve the performance of the fine-grained typing model.

Specifically, the process of the weakly-guided self-training is shown in Algorithm 1. Given the pseudo data U^l obtained from Section 4, an unlabeled dataset U, the fine-grained typing model P^e_θ, and the off-the-shelf coarse-grained NER model P^o, the training process is as follows:

For each sentence c_i in U:

- Step 1: Extract entity mentions and their coarse-grained types from c_i using the coarse-grained NER model. The output E_o is a set of $\langle x_o, t_o \rangle$ pairs, where x_o is an entity mention and t_o is a coarse-grained type. (Line 5)

- Step 2: Feed each extracted mention into the current fine-grained typing model and get the

Algorithm 1 Weakly-Guided Self-Training

```
function TRAIN($U^l$, $U$, $P_\theta^e$, $P^o$)
    while $P_\theta^e$ does not converge do
        Initialize $U' = \{\}$
        for $c_i$ in $U$ do
            $E^o = P^o(c_i)$
            for $\langle x_o, t_o \rangle$ in $E_o$ do
                $T_e = P_\theta^e(c_i, x_o)$
                Initialize $T_e' = \{\}$
                for $t_e$ in $T_e$ do
                    if If $t_e$ is a subtype of $t_o$ then
                        Add $t_e$ to $T_e'$;
                    end if
                end for
                Add $\langle c_i, x_o, T_e' \rangle$ to $U'$;
            end for
        end for
        Update $\theta$ for $P^e$ on $U'$ and $U^l$;
    end while
end function
```

fine-grained type prediction T_e. (Line 6-7)

- Step 3: Compare each predicted type $t_e \in T_e$ with t_o and consider all the subtypes of t_o in T_e as the fine-grained labels of x_o. Then we merge the new training example $\langle c_i, x_o, T_e' \rangle$ into U. (Line 8-15)

- Step 4: Update the fine-grained typing model using the expanded pseudo data. (Line 17)

6 Experiments

The experiments are conducted on two publicly available fine-grained typing tasks: FIGER (Ling and Weld, 2012) and OntoNotes Gillick et al. (2014). We evaluate our model with the following three metrics: strict accuracy, loose macro, and loose micro F-1 scores following Ling and Weld (2012).

6.1 Typing Tasks

FIGER It is sampled from Wikipedia articles and news reports (Ling and Weld, 2012). It includes about two parts: the first part used for training and development is automatically labeled from Wikipedia whereas the second part used for testing is manually labeled from news reports. For the automatically labeled data, entities are indicated by the anchor links in Wikipedia and their types are obtained by the distant supervision from Freebase. The type ontology for FIGER is extracted from Freebase and it contains two levels and 112 types in total.

OntoNotes It is obtained from Wikipedia in a similar way as FIGER dataset Gillick et al. (2014). That is, the automatically labeled data is created by distant supervision via Freebase, and manually labeled data is used as the test set. However, FIGER is overly dependent on types in the knowledge base and its types include some noise because an entity in Freebase may include multiple types but only a few of them can be inferrable from a given mention text. As a result, the automatically labeled data is further cleaned by some heuristics to filter some incorrect types which may not be deducible from the context of a given mention. Its type ontology contains three levels and 89 types in total.

6.2 Our Approach for FETw/oKB

Model Architecture Since the goal in this work is to investigate whether it is feasible to conduct FET without knowledge bases, we use the AttentiveNER model proposed by (Shimaoka et al., 2016a) and LatentNER model proposed by (Lin and Ji, 2019) as the backbone FET classifiers. In both AttentiveNER and LatentNER, the representations of the entity mention and the context are generated separately and then concatenated for final prediction. In AttentiveNER, the mention representation is the average of the embedding vectors of all the mention words. The context representation is computed by an LSTM encoder with an attention mechanism to catch the long-term dependencies. In LatentNER model, ELMo is employed as the sentence encoder. The mention representation is a weighted sum with attention. The context representation is generated from its contextualized word vectors with a mention-aware attention mechanism.

Settings Following (Shimaoka et al., 2016a), we train AttentiveNER model with the following hyperparameters: the dimension of the word embeddings $D_m = 300$; the hidden-size of the LSTM is $D_h = 100$, and the hidden-layer size of the attention module is $D_a = 1$. The optimization method is Adam(Kingma and Ba, 2014) with a learning rate of 0.005. The batch size is 1000. The context window size is $C = 15$ and the mention window size is $M = 10$. We use freely available 300-dimensional word embeddings which are trained on 840 billion tokens from the Common Crawl created by (Pennington et al., 2014) to be the only feature in our model. Following (Lin and Ji, 2019), the hyperparameters of the LatentNER model are as follows: the optimization method is Adam(Kingma and Ba, 2014) with a learning rate of 5e-5, an L2 weight decay of 0.01, a warmup rate of 0.1, and a linear learning rate decay. The batch size is 400. We use the original pretrained ELMo model (5.5B) to be the only feature in our model and its weights are frozen during training. Both the AttentiveNER model and the LatentNER model are implemented in Python using PyTorch (Paszke et al., 2019).

KB	Approach	Acc	Ma-F1	Mi-F1
No	Ours-Att	46.62	56.71	60.04
	Ours-Att+ST	54.98	69.83	69.03
	Ours-Lat	61.39	75.84	72.17
	Ours-Lat+ST	62.28	77.78	74.73
Yes	AttentiveNER	54.53	74.76	71.58
	LatentNER	62.90	83.00	79.80

Table 1: The fair comparison between approaches with and without KB on FIGER. Ours-Att and Ours-Att+ST use the same model architecture as AttentiveNER (Shimaoka et al., 2016a). Ours-Lat and Ours-Lat+ST use the same model architecture as LatentNER (Lin and Ji, 2019). Ours: trained only on pseudo data without self-training. Ours+ST: trained on pseudo data using self-training. The pseudo data is created without KB.

KB	Approach	Acc	Ma-F1	Mi-F1
No	Ours-Att	50.03	64.87	57.86
	Ours-Att+ST	52.20	68.40	61.05
	Ours-Lat	56.86	72.06	64.88
	Ours-Lat+ST	56.63	74.38	67.87
Yes	AttentiveNER	50.32	67.95	61.65
	LatentNER	54.14	73.59	67.58

Table 2: The fair comparison between approaches with and without KB on Ontonotes. Ours-Att and Ours-Att+ST use the same model architecture as AttentiveNER (Shimaoka et al., 2016a). Ours-Lat and Ours-Lat+ST use the same model architecture as LatentNER (Lin and Ji, 2019).

Implementations To implement our approaches, the pseudo data is created from a large subset of Wikipedia text, which is fully unstructured and includes about 100M sentences. Note that we do not use the anchor linking information from the original Wikipedia at all to ensure the generality of our proposed approach. For the FIGER ontology, there are 112 elements and each type corresponds to 3.98 type names on average. Then we obtain the pseudo data following the steps as described in section 4. Since the ontology in FIGER task is very similar to that in Ontonotes, the collected pseudo data can actually be re-used for the Ontonotes task by using an additional mapping from the types in FIGER ontology to the types in Ontonotes ontology. The final pseudo data contains 5.6M sentences and 7.5M mentions in total. Using the pseudo data, We train an AttentiveNER model and a LatentNER model following our proposed framework, where a coarse-grained NER model is used for weakly guided self-training, as mentioned in Section 4. Here we use a tagging model in Li et al. (2021b) for the coarse-grained NER. It is trained on the public dataset (Weischedel et al., 2013) consisting of 18 coarse types. The resultant FET models are denoted as **Ours+ST** in the following tables.

6.3 Baseline for FETw/oKB

To ensure a fair comparison, we train AttentiveNER and LatentNER on the (distantly) supervised training data provided in FIGER and Ontonotes tasks as our baseline. The hyperparameters for the baseline are exactly the same as those for our approach. This baseline is naturally served as an upper bound to some extent. In addition, we also report state-of-the-art systems on FIGER and Ontonotes tasks for reference (Ren et al., 2016a,b; Lin and Ji, 2019; Zhang et al., 2018; Chen et al., 2020; Onoe et al., 2021). Note that all these systems are trained on the supervised data as the baseline and they differ from the baseline in its sophisticated model architectures.

6.4 Experimental Results

Table 1 and Table 2 show the performance of the backbone FET models trained using our proposed approaches and those trained using the conventional framework. The approaches in the two tables are evaluated using the testing dataset of FIGER and Ontonotes, separately. Under the conventional framework (AttentiveNER and LatentNER in Table 1, 2), the FET models are trained using the original training dataset of FIGER, or Ontonotes while our approaches (Ours and Ours+ST in Table 1, 2 only use pseudo data for training. The results show that our method can be freely applied to different backbone FET models, and more advanced FET model architectures result in better performance. The performance of the self-training version of our approach (Ours-Att+ST, Ours-Lat+ST) is comparable to that of AttentiveNER or LatentNER. In addition, Ours+ST achieves a significant improvement over Ours, showing the effectiveness of the proposed self-training method. We find Ours-Lat+ST significantly outperforms LatentNER on two en-

KB	ELMo/BERT	Model	Acc	Ma-F1	Mi-F1
No	Yes	Ours+ST	62.28	77.78	74.73
Yes	No	Zhang et al. (2018)	60.2	78.7	75.5
	Yes	Lin and Ji (2019)	62.9	83.0	79.8
		Chen et al. (2020)(Exclusive)	69.1	82.6	80.8
		Chen et al. (2020)(Undefined)	65.5	80.5	78.1
		Onoe et al. (2021)	-	81.6	77.0

Table 3: The comparison between our approach in line 1 to the SOTA approaches on the FIGER dataset. Note that ours+ST is based on LatentNER while other SOTA approaches are with more sophisticated models.

KB	ELMo/BERT	Model	Acc	Ma-F1	Mi-F1
No	Yes	Ours+ST	56.63	74.38	67.87
Yes	No	Ren et al. (2016a)	55.1	71.1	64.7
		Ren et al. (2016b)	57.2	71.5	66.1
		Zhang et al. (2018)	55.52	73.33	67.61
	Yes	Lin and Ji (2019)	54.14	73.59	67.58
		Chen et al. (2020)(Undefined)	58.3	72.4	67.2
		Chen et al. (2020)(Exclusive)	58.7	73.0	68.1
		Onoe et al. (2021)	-	76.2	68.9

Table 4: The comparison between our approach in line 1 to the SOTA approaches on the Ontonotes dataset. Note that ours+ST is based on LatentNER while other SOTA approaches are with more sophisticated models.

tity types in OntoNotes: */organization/company* and */location/transit/road*. The average F1 scores of Ours-Lat+ST on these two types are 0.571 and 0.5 respectively while LatentNER achieves 0.443 and 0.4 respectively. The reason might be that these two types cover a relatively large part of the pseudo data (2.45% and 0.15% respectively) and meanwhile, the pseudo data of these two types exhibit high quality. Therefore, our FET models learn better on these two types.

As shown in the results, there is still a performance gap between ours and AttentiveNER or LatentNER. This may be mainly due to the quality of the pseudo data. The pseudo data is created from the unlabeled corpus in an automatic way without using the distant supervision from knowledge bases. Compared with the training dataset of FIGER, or Ontonotes, the pseudo data exhibits lower recall as mentioned in Section 4. In addition, there are false labels in the pseudo data. For example, in the sentence *"Travel bans imposed by the European Union have been lifted in the past in order to allow Lukashenko to attend diplomatic meetings and also to engage his government and opposition groups in dialogue."*, *Lukashenko* is mistakenly labeled as */organization/government*. Although using type names for calibration when creating type labels can preserve high precision in general, this strategy fails on this example because the type name "government" appears together with the president "Lukashenko". Therefore, the quality of the pseudo data is not as good as that of the original training dataset of FIGER or Ontonotes.

Table 3 and Table 4 report the SOTA results on these evaluation datasets. We list the SOTA results here for readers' reference.

6.5 Discussion on Complementarity

We find that the supervised training data of Ontonotes have a long-tail distribution and the LatentNER model trained under the conventional framework does not perform well on the types with a relatively small number of training examples. Table 5 shows three examples: /organization/government, /other/event, and /other/product. Luckily, our collected pseudo data have the potential to alleviate the problem of insufficient supervised data. We select the types with less than 10 examples in the training dataset of Ontonotes. The Ontonotes training dataset only contains 112 examples of the selected types while there are 1,000 times more in our pseudo data. For each of the selected types, we randomly sample 100 examples from the pseudo data and manually annotate

	Supervised Data		Pseudo Data	
type	avg. F1	prop.	# of ex.	acc.
/org./gov.	0.176	1.02%	22,680	95%
/other/event	0.286	1.68%	348,623	84%
/other/product	0.349	1.37%	80148	83%

Table 5: Analysis of the pseudo data. /org./gov.: /organization/government. prop.: proportion of the data in the KB-supervised training dataset of Ontonotes. # of ex.: the number of the examples in pseudo data. acc: accuracy. avg. F1: the average of the strict, loose-macro, loose-micro F1 scores achieved by the LatentNER model trained on the KB-supervised training dataset of Ontonotes.

if the example is correct. Using the annotations, we calculate the accuracy of the pseudo examples. The overall accuracy is 89%. Table 5 shows the accuracy of the pseudo data on the three types mentioned above. Since the collection of pseudo data is automatic and the created data exhibits high accuracy, using our method to expand the long-tailed training dataset might be an efficient way to improve the model performance.

7 Conclusion

In this work, we study a novel setting of fine-grained entity typing that disentangles type ontology from knowledge bases. This setting is more practical and provides larger flexibility for real-world applications. Under this setting, we make an initial attempt and propose a two-step framework to address FET. Experimental results show that our proposed method can achieve competitive results without accessing any KB compared to using the KB-supervised dataset for training as in the conventional framework. Although we use Shimaoka et al. (2016a) and Lin and Ji (2019) as the backbone models in our experiments, our proposed framework does not set a limit on the backbone classifier. The more advanced FET model architectures mentioned in Section 2 can be freely mitigated to our framework to achieve better performance under our setting. We leave this to future work.

8 Acknowledgement

We would like to thank anonymous reviewers for their invaluable discussions on this work. The corresponding authors are Lemao Liu and Shuming Shi.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480.

Tongfei Chen, Yunmo Chen, and Benjamin Van Durme. 2020. Hierarchical entity typing via multi-level learning to rank. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8465–8475.

Yi Chen, Haiyun Jiang, Lemao Liu, Shuming Shi, Chuang Fan, Min Yang, and Ruifeng Xu. 2021. An empirical study on multiple information sources for zero-shot fine-grained entity typing. In *EMNLP*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 87–96.

Hongliang Dai, Yangqiu Song, and Haixun Wang. 2021. Ultra-fine entity typing with weak supervision from a masked language model. *arXiv preprint arXiv:2106.04098*.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 868–878.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Coling 1992 volume 2: The 15th international conference on computational linguistics*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Anil K Jain, M Narasimha Murty, and Patrick J Flynn. 1999. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yangming Li, Lemao Liu, and Shuming Shi. 2021a. Empirical analysis of unlabeled entity problem in named entity recognition. In *Proceedings of International Conference on Learning Representations*.

Yangming Li, Lemao Liu, and Shuming Shi. 2021b. Segmenting natural language sentences via lexical unit analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Ying Lin and Heng Ji. 2019. An attentive fine-grained entity typing model with latent type representation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6198–6203.

Xiao Ling and Daniel Weld. 2012. Fine-grained entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26.

Lemao Liu, Haisong Zhang, Haiyun Jiang, Yangming Li, Enbo Zhao, Kun Xu, Linfeng Song, Suncong Zheng, Botong Zhou, Jianchen Zhu, Xiao Feng, Tao Chen, Tao Yang, Dong Yu, Feng Zhang, Zhanhui Kang, and Shuming Shi. 2021. Texsmart: A system for enhanced natural language understanding. In *The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP): System Demonstrations*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *arXiv preprint cs/0205028*.

Yukun Ma, E. Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *COLING*.

Shikhar Murty, Patrick Verga, Luke Vilnis, and Andrew McCallum. 2017. Finer grained entity typing with typenet. *arXiv preprint arXiv:1711.05795*.

Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. 2018. Hierarchical losses and new resources for fine-grained entity typing and linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 97–109.

Yasumasa Onoe, Michael Boratko, and Greg Durrett. 2021. Modeling fine-grained entity types with box embeddings. *arXiv preprint arXiv:2101.00345*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Maxim Rabinovich and Dan Klein. 2017. Fine-grained entity typing with high-multiplicity assignments. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 330–334.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1378.

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1825–1834.

Yankun Ren, J. Lin, and Jun Zhou. 2020. Neural zero-shot fine-grained entity typing. *Companion Proceedings of WWW Conference 2020*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016a. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 69–74.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016b. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1271–1280.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle

Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23.

Yongqin Xian, Christoph H. Lampert, B. Schiele, and Zeynep Akata. 2019. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:2251–2265.

Dani Yogatama, Dan Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 291–296.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2013. Hyena-live: Fine-grained online entity type classification from natural-language text. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 133–138.

Haisong Zhang, Lemao Liu, Haiyun Jiang, Yangming Li, Enbo Zhao, Kun Xu, Linfeng Song, Suncong Zheng, Botong Zhou, Jianchen Zhu, Xiao Feng, Tao Chen, Tao Yang, Dong Yu, Feng Zhang, Zhanhui Kang, and Shuming Shi. 2020. Texsmart: A text understanding system for fine-grained ner and enhanced semantic analysis. *arXiv preprint arXiv:2012.15639*.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2018. Fine-grained entity typing through increased discourse context and adaptive classification thresholds. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, *SEM@NAACL-HLT 2018*, pages 173–179.