Prompt-Learning for Fine-Grained Entity Typing

Anonymous ACL submission

Abstract

As an effective approach to tune pre-trained language models (PLMs) for specific tasks, prompt-learning has recently attracted much attention from researchers. By using *cloze*-style language prompts to stimulate the versatile knowledge of PLMs, prompt-learning can achieve promising results on a series of NLP tasks, such as natural language inference, sentiment classification, and knowledge probing. In this work, we investigate the application of prompt-learning on fine-grained entity typing in fully supervised, few-shot and zero-shot scenarios. We first develop a simple and effective prompt-learning pipeline by constructing entity-oriented verbalizer and templates and conducting masked language modeling. Further, to tackle the zero-shot regime, we propose a self-supervised strategy that carries out distribution-level optimization in prompt-learning to automatically summarize the information of entity types. Extensive experiments on three fine-grained entity typing benchmarks (with up to 86 classes) under fully supervised, few-shot and zero-shot settings show that prompt-learning methods significantly outperform fine-tuning baselines, especially when the training data is insufficient.

1 Introduction

In recent years, pre-trained language models (PLMs) have been widely explored and become a key instrument for natural language understanding (Devlin et al., 2019; Liu et al., 2019) and generation (Radford et al., 2018; Raffel et al., 2020). By applying self-supervised learning on large-scale unlabeled corpora, PLMs can capture rich lexical (Jawahar et al., 2019), syntactic (Hewitt and Manning, 2019; Wang et al., 2021), and factual knowledge (Petroni et al., 2019) that well benefits downstream NLP tasks. Considering the versatile knowledge contained in PLMs, many efforts of researchers have been devoted to stimulating task-specific knowledge in PLMs and adapting such knowledge to downstream NLP tasks. And fine-tuning with extra classifiers has been one typical solution for adapting PLMs to specific tasks in NLP tasks (Qiu et al., 2020; Han et al., 2021a).

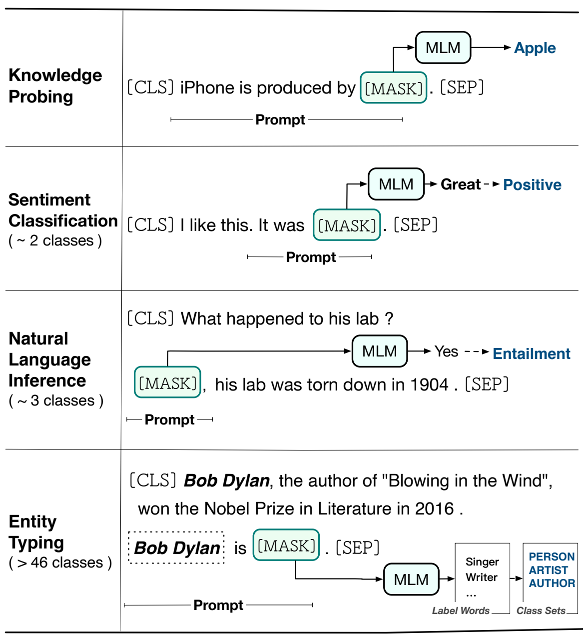


Figure 1: Examples of prompt-learning to stimulate the knowledge of PLMs by formalizing specific tasks as equivalent *cloze*-style tasks.

Some recent efforts on probing knowledge of PLMs show that, by writing some natural language prompts, we can induce PLMs to complete factual knowledge (Petroni et al., 2019). GPT-3 further utilizes the information provided by prompts to conduct few-shot learning and achieves awesome results (Brown et al., 2020). Inspired by this, prompt-learning has been introduced. As shown in Figure 1, in prompt-learning, downstream tasks are formalized as equivalent *cloze*-style tasks, and PLMs are asked to handle these tasks instead of original downstream tasks. Compared with vanilla fine-tuning methods, prompt-learning does not require extra neural layers and intuitively bridges the objective form gap between pre-training and fine-tuning. Sufficient empirical analysis shows that, either for manually picking hand-crafted prompts (Liu et al., 2021; Han et al., 2021b) or automatically building auto-generated prompts (Gao et al., 2020; Lester et al., 2021), taking prompts for tuning models is surprisingly effective for the knowledge stimulation and model adaptation of PLMs, especially in the low-data regime (Ding et al., 2021a).

Intuitively, prompt-learning is applicable to fine-grained entity typing, which aims at classifying marked entities from input sequences into specific types in a pre-defined label set. We discuss this topic with a motivating example, "*He is from New York*". By adding a prompt with a masking token `[MASK]`, the sentence becomes "*He is from New York. In this sentence, New York is* `[MASK]`". Due to the wealth of knowledge acquired during pre-training, PLMs can compute a probability distribution over the vocabulary at the masked position, and a relatively higher probability with the word "city" than the word "person". In other words, with simple prompts, the abstract entity attributes contained in PLMs can be efficiently exploited, which is meaningful for downstream entity-related tasks.

In this work, we comprehensively explore the application of prompt-learning to fine-grained entity typing in fully supervised, few-shot and zero-shot settings. Particularly, we first introduce a naive pipeline, where we construct entity-oriented prompts and formalize fine-grained entity typing as a *cloze*-style task. This simple pipeline yields promising results in our experiments, especially when supervision is insufficient. Then, to tackle the zero-shot scenario where no explicit supervision exists in training, we develop a self-supervised strategy under our prompt-learning pipeline. Our self-supervised strategy attempts to automatically summarize entity types by optimizing the similarity of the predicted probability distributions of paired examples in prompt-learning.

Three popular benchmarks are used for our experiments, including FEW-NERD (Ding et al., 2021c), OntoNotes (Weischedel et al., 2013), BBN (Weischedel and Brunstein, 2005). All these datasets have a complex type hierarchy consisting of rich entity types, requiring models to have good capabilities of entity attribute detection. Empirically, our method yields significant improvements on these benchmark datasets, especially under the zero-shot and few-shot settings. We also make an analysis and point out both the superiority and bottleneck of prompt-learning in fine-grained entity typing, which may advance further efforts to extract entity attributes using PLMs. Our source code and pre-trained models will be publicly available.

2 Background

In this section, we first give a problem definition of the entity typing task (§ 2.1), followed by an introduction of conventional vanilla fine-tuning (§ 2.2) and prompt-based tuning (§ 2.3) with PLMs.

2.1 Problem Definition

The input of entity typing is a dataset $\mathcal{D} = \{x_1, ..., x_n\}$ with n sentences, and each sentence x contains a marked entity mention m. For each input sentence x, entity typing aims at predicting the entity type $y \in \mathcal{Y}$ of its marked mention m, where $\mathcal{Y}$ is a pre-defined set of entity types. Entity typing is typically regarded as a context-aware classification task. For example, in the sentence "*London* is the fifth album by the rock band Jesus Jones...", the entity mention *London* should be classified as `Music` rather than `Location`. Using pre-trained neural language models (e.g. BERT) as the encoder and performing model tuning for classifying types becomes a standard paradigm in recent years.

2.2 Vanilla Fine-tuning

In the vanilla fine-tuning paradigm of entity typing, for each token t_i in an input sequence $x = \{\texttt{[CLS]}, t_1, \ldots, m, \ldots, t_T, \texttt{[SEP]}\}$ with a marked entity mention $m = \{t_i, \ldots, t_j\}$, the PLM $\mathcal{M}$ produces its contextualized representation $\{\mathbf{h}_{\texttt{[CLS]}}, \mathbf{h}_1, \ldots, \mathbf{h}_T, \mathbf{h}_{\texttt{[SEP]}}\}$. Empirically, we choose the embedding of the `[CLS]` token, $\mathbf{h}_{\texttt{[CLS]}}$, as the final representation that is fed into an output layer to predict the probability distribution over the label space

$$P(y \in \mathcal{Y}|s) = \texttt{softmax}(\mathbf{W}\mathbf{h}_{\texttt{[CLS]}} + \mathbf{b}), \quad (1)$$

where $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters. $\mathbf{W}$, $\mathbf{b}$ and all parameters of PLMs are tuned by maximizing the objective function $\frac{1}{n}\sum_{i=1}^{n}\log(P(y_i|s_i))$, where y_i is the golden type label of s_i.

2.3 Prompt-based Tuning

In prompt-based tuning, for each label $y \in \mathcal{Y}$, we define a label word set $\mathcal{V}_y = \{w_1, \ldots, w_m\}$. $\mathcal{V}_y$ is a subset of the vocabulary $\mathcal{V}$ of the PLM $\mathcal{M}$,

Figure 2: The illustration of prompt-learning for fine-grained entity typing with supervision. We take hard-encoding prompt strategy as an example in this figure.

i.e., $\mathcal{V}_y \subseteq \mathcal{V}$. By taking the union of the dictionary corresponding to each label, we get an overall dictionary $\mathcal{V}^*$. For example, in sentiment classification, we could map the label y = POSITIVE into a set $\mathcal{V}_y = \{$*great, good, wonderful...*$\}$. And another primary component of prompt-learning is a prompt template $T(\cdot)$, which modifies the original input x into a prompt input $T(x)$ by adding a set of additional tokens at the end of x. Conventionally, a [MASK] token is added for PLMs to predict the missing label word $w \in \mathcal{V}^*$. Thus, in prompt-learning, a classification problem is transferred into a masked language modeling problem,

$$p(y \in \mathcal{Y}|s) = p([\texttt{MASK}] = w \in \mathcal{V}_y | T(s)). \quad (2)$$

3 Prompt-learning for Entity Typing: A Naive Pipeline

After transferred into masked language modeling, the prompt-learning method is applicable to learning and aggregating type information of entities. In this section, we first introduce a naive but empirically strong baseline that utilizes prompts to extract entity types with explicit supervision, including the construction of label words (§ 3.1), templates (§ 3.2) and training (§ 3.3). And such a simple pipeline yields remarkable results on three benchmark datasets. Then we propose a self-supervised prompt-learning method that automatically learns type information from unlabeled data (§ 4).

3.1 Label Words Set $\mathcal{V}^*$

For fine-grained entity typing, datasets usually use hierarchical label space such as PERSON/ARTIST (FEW-NERD) and ORGANIZATION/PARTY (OntoNotes). In this case, we use all the words as the label words set $\mathcal{V}^*$ for this entity type. For example, y = LOCATION/CITY $\rightarrow$ $v = \{$*location, city*$\}$. And as the entity types are all well-defined nouns with clear boundaries, it is intuitive to expand the label words set $\mathcal{V}^*$ with obtainable related nouns. For example, in Related Words[1], the top-5 related words of the label word *city* is "*metropolis*, *town*, *municipality*, *urban*, *suburb*". These words are strongly related to the class CITY, and they are hardly mapped to other entity types even under the same LOCATION class, such as LOCATION/MOUNTAIN or LOCATION/ISLAND.

In masked language modeling, we use confidence scores of all the words in $\mathcal{V}_y$ to construct the final score of the particular type y. That is, for an input x (which is mapped to $T(x)$) and its entity type y (which is mapped to $\mathcal{V}_y = \{w_1, ..., w_m\}$), the conditional probability becomes

$$P(y|x) = \frac{1}{m}\sum_{j}^{m} \lambda_j P([\texttt{MASK}] = w_j | T(x)), \quad (3)$$

where λ_i is a parameter to indicate the importance of the current word $w_j \in \mathcal{V}_y$. Note that λ_i could also be learnable or heuristically defined.

3.2 Templates

In this section, we construct entity-oriented prompts for the fine-grained entity typing task. We choose hard-encoding templates with natural language and soft-encoding templates with additional special tokens in our work.

For the choice of hard-encoding templates, we do not use automatic searching methods for discrete prompts since the fine-grained entity typing task is clearly defined and the prompts are easily purposeful. We select simple declarative templates rather than hypernym templates to avoid grammatical errors. In the template of hard encoding setting, we first copy the marked entity mention in x, then we add a few linking verbs and articles followed by the [MASK] token. With the marked entity mention

[1] https://relatedwords.org

[Ent], we use the following templates:

$$\mathrm{T}_1(x) = x.\ \texttt{[Ent]}\ \text{is}\ \texttt{[MASK]},$$

$$\mathrm{T}_2(x) = x.\ \texttt{[Ent]}\ \text{is a}\ \texttt{[MASK]},$$

$$\mathrm{T}_3(x) = x.\ \text{In this sentence,}\ \texttt{[Ent]}\ \text{is a}\ \texttt{[MASK]},$$

where [Ent] is the entity mention in x. In § 5, we report the the results of $\mathrm{T}_3(\cdot)$.

We also adopt the soft-encoding strategy, which introduces some additional special tokens $[\mathrm{P}_1], ..., [\mathrm{P}_l]$ as the template, where l is a pre-defined hyper-parameter. The template begins with a delimiter [P] and a copy of the entity mention [M]. The complete template becomes:

$$\mathrm{T}_4(x) = x\ \texttt{[P]}\ \texttt{[Ent]}\ [\mathrm{P}_1],...,\ [\mathrm{P}_l]\ \texttt{[MASK]},$$

where each embedding of prompts is randomly initialized and optimized during training. Intuitively, these special tokens can represent a cluster of words with similar semantics in the vocabulary.

3.3 Training and Inference

The strategies of hard or soft encoding provide different initialization of templates, and they both can be parameterized by ϕ and optimized along with $\mathcal{M}$ during training. We train the pre-trained model $\mathcal{M}$ (parameterized by θ) along with the additional prompt embeddings by the cross-entropy loss:

$$\mathcal{L} = -\sum \log P(y|x;\theta,\phi). \tag{4}$$

For inference, we can directly use Eq. 3 to predict the label of the current input instance based on the predicted words of the [MASK] position.

This pipeline could be applied to entity typing with explicit supervision, and it is more effective when the training data are insufficient, i.e., the few-shot scenario (§ 5.3). Naturally, we take further step and consider a more extreme situation, that is, a scenario without any training data (zero-shot scenario). In this setting, if we directly use an additional classifier to predict the label, the result is equivalent to random guessing since the parameters of the classifier are randomly initialized. If we use prompts to infer the label based on the predicted words, although its performance is significantly better than guessing, there will also be a catastrophic decline (§ 5.4). To this end, a question emerges: *"Is it possible for PLMs to predict entity types without any explicit supervision? "*

4 Self-supervised Prompt-learning for Zero-shot Entity Typing

With prompt-learning, the answer is yes, because in the pre-training stage, the contexts of entities have already implied the corresponding type information, which provides an advantageous initialization point for the prompt-learning paradigm. For example, in the input sentence with the $T_3(\cdot)$ template: "Steve Jobs found Apple. In this sentence, Steve Jobs is a [MASK] ". In our observations, the probability of PLMs predicting *person* at the masked position will be significantly higher than the probability of *location*. And if we make reasonable use of this superior initialization point, it is possible for PLMs to automatically summarize the type information, and finally extract the correct entity type.

4.1 Overview

In order to create conditions for PLMs to summarize entity types, we consider a self-supervised paradigm that optimizes the similarity of the probability distribution predicted by similar examples over a projected vocabulary $\mathcal{V}^*$. To achieve that in prompt-learning, we need to (1) impose a limit on the prediction range of the model, so that only those words that we need, that is, words that express entity types, participate in the optimization of the gradient; (2) provide an unlabeled dataset, where entity mentions are marked without any types to allow the model to learn the process of inducing type information in a self-supervised manner. The inputs contain a pre-trained model $\mathcal{M}$, a pre-defined label schema $\mathcal{Y}$, and a dataset without labels $\mathcal{D} = \{x_1, ..., x_n\}$ (entity mentions are marked without any types). our goal is to make $\mathcal{M}$ capable to automatically carry out zero-shot entity typing after trained on $\mathcal{D}$ and $\mathcal{Y}$. Using prompt-learning as the training strategy, we first construct a label words set $\mathcal{V}^*$ from $\mathcal{Y}$, and for each sentence x in $\mathcal{D}$, we wrap it with hard-encoding template with a [MASK] symbol. The key idea is to make the prediction distributions of the same type of entities on $\mathcal{V}^*$ as similar as possible. In this way, we can perform contrastive learning by sampling positive and negative examples, while ignoring the impact of other words that are not in $\mathcal{V}^*$ on optimization during the MLM process.

4.2 Self-supervised Learning

Although there are no labels in $\mathcal{D}$, we can still develop a sampling strategy based on a simple

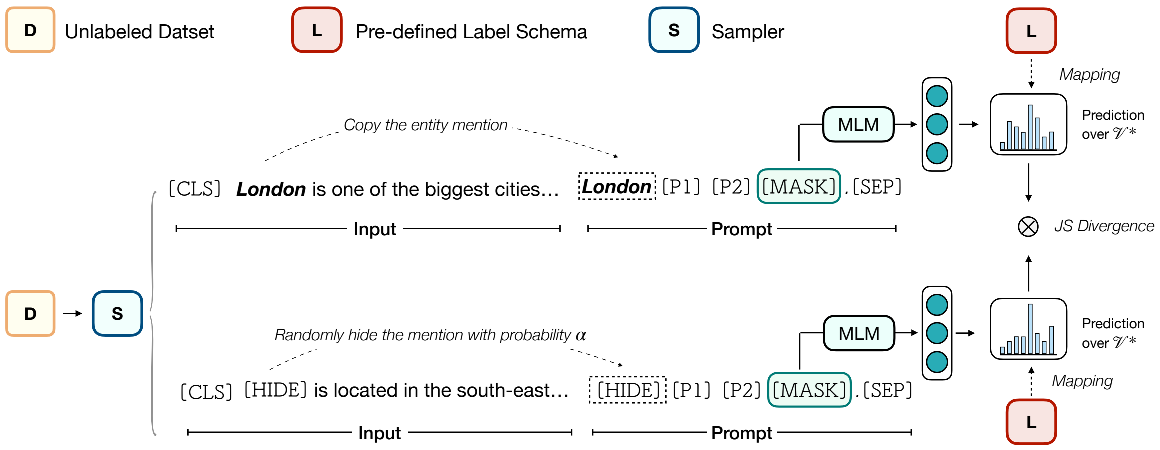


Figure 3: The illustration of self-supervised prompt-learning for fine-grained entity typing with unlabeled data and a pre-defined label set. $\mathcal{V}^*$ denotes the label words projected from the input label set. Note that we only show the positive pair in this figure.

hypothesis, that is, *same entities in different sentences have similar types*. For instance, we will sample two sentences contain "Steve Jobs" as a positive pair. Moreover, considering entity typing is context-aware, "Steve Jobs" could be *entrepreneur*, *designer*, *philanthropist* in different contexts, we choose to optimize the similarity between distributions of the words over $\mathcal{V}^*$. This strategy not only softens the supervision, but also eliminates the impact of other words in self-supervised learning.

Particularly, we randomly sample c positive pairs, i.e., sentence pairs that share one same entity mention, denoted as $\hat{\mathcal{D}}_{\text{pos}}$, and c negative pairs, i.e., two sentences with different entity mentions marked, denoted as $\hat{\mathcal{D}}_{\text{neg}}$ from a large-scale entity-linked corpus $\mathcal{D}$. To avoid generating false negative samples, the negative samples are further restricted by a large dictionary that contains common entities and their type information. Only sentence pairs with entities of different types in the dictionary are selected as negative samples. Then we wrap them with hard-encoding $\mathrm{T}_3(\cdot)$. To avoid overfitting of the entity names, we randomly hide the entity mention (in the original input and the template) with a special symbol [Hide] with a probability of α. Empirically, α is set to 0.4.

Since the impact of a pair of examples on training should be measured at the distribution level, we choose Jensen-Shannon divergence as a metric to assess the similarity of two distributions. Thus, in a sentence pair (x, x'), the similarity score of two representations of the the predictions $\mathbf{h}$ and $\mathbf{h}'$ of the [MASK] position is computed by:

$$s(\mathbf{h}, \mathbf{h}') = \mathrm{JS}(P_{\mathcal{V}^*}(w|x), P_{\mathcal{V}^*}(w|x')), \tag{5}$$

where JS is Jensen-Shannon divergence, $P_{\mathcal{V}^*}(w|x)$ and $P_{\mathcal{V}^*}(w|x')$ are probability distributions of the predicting token w over $\mathcal{V}^*$ obtained by $\mathbf{h}$ and $\mathbf{h}'$.

As we attempt to make the predictions of the positive pairs similar, the objective is computed by:

$$\begin{aligned}\mathcal{L} = & -\frac{1}{|\hat{\mathcal{D}}_{\text{pos}}|^2} \sum_{x \in \hat{\mathcal{D}}_{\text{pos}}} \sum_{x' \in \hat{\mathcal{D}}_{\text{pos}}} \log(1 - s(\mathbf{h}, \mathbf{h}')) \\ & - \frac{1}{|\hat{\mathcal{D}}_{\text{neg}}|^2} \sum_{x \in \hat{\mathcal{D}}_{\text{neg}}} \sum_{x' \in \hat{\mathcal{D}}_{\text{neg}}} \gamma \log(s(\mathbf{h}, \mathbf{h}')),\end{aligned} \tag{6}$$

where γ is a penalty term because the assumption is loose in negative pairs. We use entity-linked Wikipedia corpus as the raw data and generate about 1 million pairs of data each as $\hat{\mathcal{D}}_{\text{pos}}$ and $\hat{\mathcal{D}}_{\text{neg}}$.

5 Experiments

We evaluate our methods on three widely used fine-grained entity typing datasets, the dataset split and experimental details are reported in Appendix A.

5.1 Datasets

We use the following three fine-grained entity typing datasets in our experiments.

FEW-NERD We use FEW-NERD (Ding et al., 2021c) as the main dataset, which has the following advantages: (1) FEW-NERD is large-scale and fine-grained, which contains 8 coarse-grained and 66 fine-grained entity types. (2) FEW-NERD is manually annotated, thereby we can precisely assess the capability of entity typing models. We use the supervised setting of the dataset, FEW-NERD (SUP), and the official split in our experiments.

Shot	Metric	Few-NERD		OntoNotes		BBN	
		Fine-tuning	PLET	Fine-tuning	PLET	Fine-tuning	PLET
1	Acc	8.94	43.87 (+34.93)	3.70	38.97 (+35.27)	0.80	40.70 (+39.90)
	MiF	19.85	60.60 (+45.75)	18.98	59.91 (+40.93)	5.79	49.25 (+43.46)
	MaF	19.85	60.60 (+40.75)	19.43	61.42 (+41.99)	4.42	48.48 (+43.06)
2	Acc	20.83	47.78 (+26.95)	7.27	39.19 (+31.92)	6.68	41.33 (+34.65)
	MiF	32.67	62.09 (+29.42)	24.89	61.09 (+36.20)	13.70	54.00 (+40.30)
	MaF	32.67	62.09 (+29.42)	25.64	62.68 (+37.04)	13.23	51.97 (+38.74)
4	Acc	33.09	57.00 (+23.91)	11.15	38.39 (+27.24)	19.34	52.21 (+32.87)
	MiF	44.14	68.61 (+24.47)	27.69	59.81 (+32.12)	27.03	61.13 (+34.10)
	MaF	44.14	68.61 (+24.47)	28.26	60.89 (+32.63)	24.69	58.91 (+34.22)
8	Acc	46.44	55.75 (+9.31)	18.37	39.37 (+21.00)	27.01	44.30 (+17.29)
	MiF	57.76	68.74 (+10.98)	38.16	57.97 (+19.81)	40.19	56.21 (+16.02)
	MaF	57.76	68.74 (+10.98)	37.77	58.32 (+20.55)	39.50	55.15 (+15.65)
16	Acc	60.98	61.58 (+0.60)	32.26	42.29 (+10.03)	39.67	55.00 (+15.33)
	MiF	71.59	72.39 (+0.80)	51.40	60.79 (+9.39)	49.01	62.84 (+13.83)
	MaF	71.59	72.39 (+0.80)	51.45	61.80 (+10.35)	47.09	62.38 (+15.29)

Table 1: Results of few-shot entity typing on FEW-NERD, OntoNotes and BBN, all the methods use $\text{BERT}_{\text{base}}$ with same initialization weights as the backbone encoder. Training set and dev set have the same size.

OntoNotes We also use the OntoNotes 5.0 dataset (Weischedel et al., 2013). Following previous works for fine-grained entity typing, we adopt 86-classes version of OntoNotes, while each class has at most 3 levels of the type hierarchy. And the data split is identical to (Shimaoka et al., 2017).

BBN. BBN dataset is selected from Penn Treebank corpus of Wall Street Journal texts and labeled by (Weischedel and Brunstein, 2005). We follow the version processed by (Ren et al., 2016a), and the data split by (Ren et al., 2016b). The dataset contains 46 types and each type has a maximum type hierarchy level of 2.

5.2 Results of Fully Supervised Entity Typing

Dataset	Metric	Method		
		FT	PLET (H)	PLET (S)
Few-NERD	Acc	79.75	**79.90**	79.86
	MiF	85.74	**85.84**	85.76
	MaF	85.74	**85.84**	85.76
OntoNotes	Acc	59.71	60.37	**65.68**
	MiF	70.47	70.78	**74.53**
	MaF	76.57	76.42	**79.77**
BBN	Acc	62.39	**65.92**	63.11
	MiF	68.88	**71.55**	68.68
	MaF	67.37	**70.82**	67.81

Table 2: Fully supervised entity typing results. FT denotes the vanilla fine-tuning method, (H) denotes the hard-encoding and (S) denotes the soft-encoding.

The results on all three datasets across different models are reported in Table 2. Overall, the prompt-based methods have shown certain improvements comparing to directly fine-tuned models. It shows that the prompt-based method does help with capturing entity-type information from a given context.

It is also observed that the magnitude of the improvement and the preference of prompt encoding strategy may vary with different datasets. The prompt-based method seems less effective on FEW-NERD dataset than the other two. It indicates that the effect of the prompt-based method partially depends on the characteristics of the dataset and that different prompt designs may suit different data. Specifically, FEW-NERD is manually annotated and contains much less noise than the other two datasets, benefiting the FT method to learn classification with an extra linear layer. Moreover, for the OntoNotes dataset, soft encoding significantly outperforms hard encoding, while for the other two datasets the effect seems reversed.

5.3 Results of Few-shot Entity Typing

Table 1 shows the results on few-shot entity typing. It is shown that prompt-based model outperforms fine-tuning by a large margin under few-shot setting, especially when only $1 \sim 2$ training instances per type are available. It should be noted that for OntoNotes and BBN datasets, sampling 16 instances for each entity type already amounts to over 0.5% of the total training data. Meanwhile, some of the data in BBN are distantly-supervised

Dataset	Metric	Method	
		PLET	PLET (S)
Few-NERD	Acc	17.55	23.99 (+6.44)
	MiF	28.39	47.98 (+19.59)
	MaF	28.39	47.98 (+19.59)
OntoNotes‡	Acc	25.10	28.27 (+3.17)
	MiF	33.61	49.79 (+16.18)
	MaF	37.91	49.95 (+12.04)
BBN	Acc	55.82	57.79 (+1.97)
	MiF	60.64	63.24 (+2.60)
	MaF	59.99	64.00 (+4.01)

Table 3: Results of zero-shot entity typing on FEW-NERD, OntoNotes, and BBN. ‡ means that we remove the "Other" class during testing. PLET denotes the prompt-learning pipeline and PLET (S) denotes self-supervised prompt-learning.

and are potentially erroneous. It brings more randomness to few-shot training. The results support the idea that a well-designed prompt has much potential in mining the learned knowledge in pre-trained models and thus yields better performance in few-shot settings. The results also indicate that even when the number of entity types is large (46 $\sim$ 86), the superiority of prompt-learning still holds.

5.4 Results of Zero-shot Entity Typing

Table 3 shows the results on zero-shot entity typing task on FEW-NERD dataset. We did not report the performance of the vanilla fine-tuning approach because it cannot produce reasonable results with a randomly initialized classifier. And it also should be noted that the prompt method without fine-tuning already outperforms random guessing. It indicates that adding a prompt is informative for a model pre-trained on masked-language-model task (e.g. BERT) and can induce reasonable predictions in entity typing tasks. Second, the performance of the model improves by a large margin if trained on unlabeled data. It shows the effectiveness of the proposed self-supervised training approach and points to the potential of a pre-trained prompt-based model under the zero-shot setting when no labeled data are available.

To explore the more subtle changes in performance, we carry out case study for the zero-shot entity typing. In Figure 4, we illustrate the zero-shot prediction distribution (the correct prediction and other top-5 predictions) for four entity types in FEW-NERD. We could observe that with self-supervised prompt-learning, PLET (S) could summarize entity type information and infer the related words to a certain extent. In Figure 4 (a) and Figure 4 (b), the PLET model suffers from a severe bias and almost predict no correct labels in the zero-shot setting since such words are low-frequency. And although there is no explicit supervision in the pre-training stage of UNPLET, the model could still find the corresponding words that express the ORG-SPORTSLEAGUE and the EVENT-ATTACK types. In Figure 4 (c), self-supervised learning increases the performance of the original encoder. Further, in Figure 4 (d), PLET has been able to make satisfying predictions for this type LOC-MOUNTAIN. In this case, the use of self-supervised learning has hardly weakened the performance, which means that the process of automatically summarizing type information has little negative impact on high-confidence entity types.

Type	Template	Acc	MiF	MaF
Hard	$T_1(x)$	54.45	67.34	67.34
	$T_2(x)$	53.93	66.44	66.44
	$T_3(x)$	**55.75**	**68.74**	**68.74**
Soft	l = 2	**59.25**	**69.58**	**69.58**
	l = 3	53.66	66.06	66.06
	l = 4	52.96	66.01	66.01
	l = 5	55.44	68.39	68.39

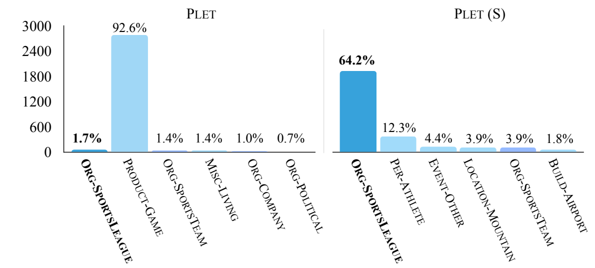
Table 4: Effect of templates. The results are produced under 8-shot setting on FEW-NERD dataset by PLET. l is the number of soft tokens.

5.5 Effect of Templates

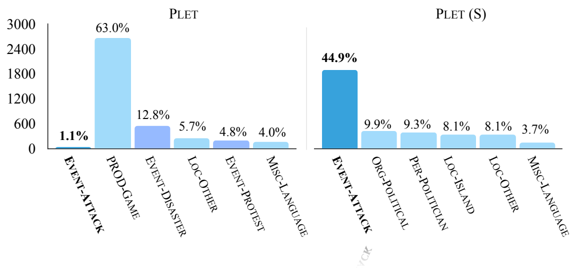
As stated in previous studies (Zhao et al., 2021), the choice of templates may have a huge impact on the performance in prompt-learning, We carry out experiments under the 8-shot setting on FEW-NERD dataset to investigate such influence. And we use 3 different hard templates and 4 soft templates (by changing the number of prompt tokens l). The results demonstrate that the choice of templates exerts a considerable influence on the performance of prompt-based few-shot learning. For the hard templates, the phrase that describes the location "in this sentence" contributes a remarkable improvement in performance. For the soft templates, surprisingly, the prompt-learning model yields the best result with the fewest special tokens.

6 Related Work

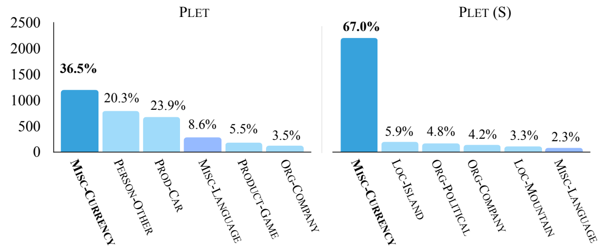
After a series of effective PLMs like GPT (Radford et al., 2018) and BERT (Devlin et al., 2019), fine-tuned PLMs have demonstrated their effectiveness



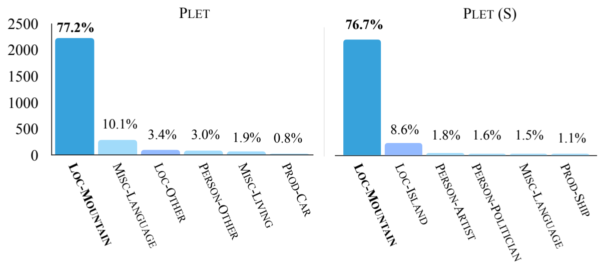
(a) Zero-shot prediction distribution on ORG-SPORTSLEAGUE.



(b) Zero-shot prediction distribution on EVENT-ATTACK.



(c) Zero-shot prediction distribution on MISC-CURRENCY.



(d) Zero-shot prediction distribution on LOC-MOUNTAIN.

Figure 4: Zero-shot prediction distribution on 4 types in FEW-NERD. In each subgraph, the left part illustrates the results of PLET and the right part are PLET (S). ■ denotes the correct predictions, ■ denotes the wrong predictions with correct coarse-grained types, and ■ denotes the wrong predictions with wrong coarse-grained types.

on various important NLP tasks (Baldini Soares et al., 2019; Peng et al., 2020; Ding et al., 2021b).

Despite the success of fine-tuning PLMs, the huge objective form gap between pre-training and fine-tuning still hinders the full use of per-trained knowledge for downstream tasks (Liu et al., 2021; Han et al., 2021b; Hu et al., 2021). To this end, prompt-learning has been proposed. The seminal work that stimulates the development of prompt-learning is the birth of GPT-3 (Brown et al., 2020), which uses hand-crafted prompts for tuning and achieves impressive performance on various tasks. A series of hand-crafted prompts have been widely explored in knowledge probing (Petroni et al., 2019; Davison et al., 2019), relation classification (Han et al., 2021b), sentiment classification and natural language inference (Schick and Schütze, 2021; Liu et al., 2021). To avoid labor-intensive prompt design, automatic prompt search has also been extensively explored Schick et al. (2020); Schick and Schütze (2021); Shin et al. (2020); Gao et al. (2020) to generate language phrases for prompts. Recently, some continuous prompts have also been proposed (Li and Liang, 2021; Lester et al., 2021), which directly use a series of learnable continuous embeddings as prompts rather than discrete language phrases.

This paper aims to stimulate PLMs with prompt-learning to capture the attribute information of entities. We take fine-grained entity typing, a crucial task in knowledge extraction to assign entity types to entity mentions (Lin et al., 2012), as the foothold to develop prompt-learning strategies. In fact, Dai et al. (2021) use hypernym extraction patterns to enhance the context and apply masked language modeling to tackle the ultra-fine entity typing problem (Choi et al., 2018) with free-form labels, which shares a similar intuition with prompt-learning. In our work, we mainly emphasize using prompt-learning to extract entity types that have been pre-defined in low-data scenarios.

7 Conclusion

This work investigates the application of prompt-learning on fine-grained entity typing in in fully supervised, few-shot and zero-shot scenarios. We first investigate a simple and effective prompt-learning pipeline that could be used to extract entity types with both sufficient and insufficient supervision. Furthermore, to handle the zero-shot setting, we propose a self-supervised prompt-learning approach that automatically learns and summarizes entity types based on unlabeled corpora and a pre-defined label schema, which utilizes prompts to take advantage of prior knowledge distributed in PLMs, and could learn pre-defined type information without overfitting by performing distribution-level optimization.

References

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Proceedings of NIPS*, pages 1877–1901.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of ACL*, pages 87–96.

Hongliang Dai, Yangqiu Song, and Haixun Wang. 2021. Ultra-fine entity typing with weak supervision from a masked language model. In *Proceedings of ACL*, pages 1790–1799.

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of EMNLP-IJCNLP*, pages 1173–1178.

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

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. 2021a. Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998*.

Ning Ding, Xiaobin Wang, Yao Fu, Guangwei Xu, Rui Wang, Pengjun Xie, Ying Shen, Fei Huang, Hai-Tao Zheng, and Rui Zhang. 2021b. Prototypical representation learning for relation extraction. In *Proceedings of ICLR*.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Hai-Tao Zheng, and Zhiyuan Liu. 2021c. Few-nerd: A few-shot named entity recognition dataset. In *Proceedings of ACL*, pages 3198–3213.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, et al. 2021a. Pre-trained models: Past, present and future. *arXiv preprint arXiv:2106.07139*.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021b. Ptr: Prompt tuning with rules for text classification. *arXiv preprint arXiv:2105.11259*.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of NAACL*, pages 4129–4138.

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Juanzi Li, and Maosong Sun. 2021. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. *arXiv preprint arXiv:2108.02035*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *Proceedings of ACL*, pages 3651–3657.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Thomas Lin, Oren Etzioni, et al. 2012. No noun phrase left behind: detecting and typing unlinkable entities. In *Proceedings of EMNLP-CoNLL*, pages 893–903.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *AAAI*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of ICLR*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of NIPS*, pages 8024–8035.

Hao Peng, Tianyu Gao, Xu Han, Yankai Lin, Peng Li, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2020. Learning from Context or Names? An Empirical Study on Neural Relation Extraction. In *Proceedings of EMNLP*, pages 3661–3672.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of EMNLP*, pages 2463–2473.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21:1–67.

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, Austin, Texas. Association for Computational Linguistics.

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 SIGKDD*, page 1825–1834.

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of COLING*, pages 5569–5578.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of EACL*, pages 255–269.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1271–1280, Valencia, Spain. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models using automatically generated prompts. In *Proceedings of EMNLP*, pages 4222–4235.

Dong Wang, Ning Ding, Piji Li, and Haitao Zheng. 2021. CLINE: Contrastive learning with semantic negative examples for natural language understanding. In *Proceedings of ACL*.

Ralph Weischedel and Ada Brunstein. 2005. BBN Pronoun Coreference and Entity Type Corpus. Linguistic Data Consortium, Philadelphia.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. OntoNotes Release 5.0. Abacus Data Network.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP*, pages 38–45.

Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*.

A Experimental Settings and Details

A.1 Experimental Settings

The experiments are performed under three different settings to evaluate the effect of the prompt-learning method and semi-supervised training. In table 5, we show the statistics of all the settings on the three datasets.

Supervised Setting. In a fully supervised setting, all training data are used in the training phase. FT and PLET are used to train the model. We run the experiments on all three datasets with BERT-base-cased backbone. Both hard and soft encodings are used for PLET.

Few-shot Setting. In a few-shot setting, we randomly sample 1, 2, 4, 8, 16 instances for each entity type for training. We apply both FT and PLET methods with hard encoding on all the three datasets.

Zero-shot Setting. In zero-shot setting, no training data with labels are available. The model is required to infer the entity type without any supervised training. Since fine-tuning is not applicable in this setting, we only conduct experiments on PLET and PLET (S).

Metrics. In terms of evaluation metrics, we follow the widely used setting of Ling and Weld (2012), which includes strict accuracy (Acc), loose macro F1-score (MaF) and loose micro F1-score (MiF) to evaluate the performances of models. The loose F1-score calculation concerns type labels by different granularities.

A.2 Experimental Details

We use BERT-base (Devlin et al., 2019) as the backbone structures of our model and initialized with the corresponding pre-trained cased weights[2]. The hidden sizes are 768, and the number of layers are 12. Models are implemented by Pytorch framework[3] (Paszke et al., 2019) and Huggingface transformers[4] (Wolf et al., 2020). BERT models are optimized by AdamW (Loshchilov and Hutter, 2019) with the learning rate of 5e-5. The training batch size used is 16 for all models. In the supervised setting, each model is trained for 10 epochs and evaluated on the dev set every 2000 steps. In the few-shot setting, each model is trained for 30 epochs and evaluated every 10∼50 steps, each time the evaluation is run for 200 steps. For the methods with hard-encoding, we report the experimental results of $T_3(\cdot)$. For the soft-encoding method, we report the results of $m = 2$. Experiments are conducted with CUDA on NVIDIA Tesla V100 GPUs.

[2] https://github.com/google-research/bert

[3] https://pytorch.org

[4] https://github.com/huggingface/transformers

Dataset	#Type	Supervised			Few-shot			Zero-shot		
		$\lvert\mathcal{D}_{\text{train}}\rvert$	$\lvert\mathcal{D}_{\text{dev}}\rvert$	$\lvert\mathcal{D}_{\text{test}}\rvert$	$\lvert\mathcal{D}_{\text{train}}\rvert$	$\lvert\mathcal{D}_{\text{dev}}\rvert$	$\lvert\mathcal{D}_{\text{test}}\rvert$	$\lvert\mathcal{D}_{\text{train}}\rvert$	$\lvert\mathcal{D}_{\text{dev}}\rvert$	$\lvert\mathcal{D}_{\text{test}}\rvert$
Few-NERD	66	340,382	48,758	96,901	66~1,056	$= \lvert\mathcal{D}_{\text{train}}\rvert$	96,901	0	0	96,901
OntoNotes	86	253,239	2,200	8,962	86~1,376	$= \lvert\mathcal{D}_{\text{train}}\rvert$	8,962	0	0	8,962
BBN	46	86,077	12,824	12,824	46~736	$= \lvert\mathcal{D}_{\text{train}}\rvert$	12,824	0	0	12,824

Table 5: Statistics of FEW-NERD, OntoNotes and BBN from three experimental settings. For all three settings, the test sets are identical. For the training set of the few-shot setting, we report the summation from 1-shot to 16-shot.