information

Article

Decomposed Two-Stage Prompt Learning for Few-Shot Named Entity Recognition

Feiyang Ye [1], Liang Huang [2,*], Senjie Liang [1] and KaiKai Chi [2]

[1] The College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China; feiyangye@zjut.edu.cn (F.Y.); senjieliang@zjut.edu.cn (S.L.)
[2] The College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China; kkchi@zjut.edu.cn
* Correspondence: lianghuang@zjut.edu.cn

Abstract: Named entity recognition (NER) in a few-shot setting is an extremely challenging task, and most existing methods fail to account for the gap between NER tasks and pre-trained language models. Although prompt learning has been successfully applied in few-shot classification tasks, adapting to token-level classification similar to the NER task presents challenges in terms of time consumption and efficiency. In this work, we propose a decomposed prompt learning NER framework for few-shot settings, decomposing the NER task into two stages: entity locating and entity typing. In training, the location information of distant labels is used to train the entity locating model. A concise but effective prompt template is built to train the entity typing model. In inference, a pipeline approach is used to handle the entire NER task, which elegantly resolves time-consuming and inefficient problems. Specifically, a well-trained entity locating model is used to predict entity spans for each input. The input is then transformed using prompt templates, and the well-trained entity typing model is used to predict their types in a single step. Experimental results demonstrate that our framework outperforms previous prompt-based methods by an average of 2.3–12.9% in F1 score while achieving the best trade-off between accuracy and inference speed.

Keywords: named entity recognition; prompt learning; few-shot; distant labels; natural language processing

Citation: Ye, F.; Huang, L.; Liang, S.; Chi, K. Decomposed Two-Stage Prompt Learning for Few-Shot Named Entity Recognition. *Information* **2023**, *14*, 262. https://doi.org/10.3390/info14050262

Academic Editor: Kostas Stefanidis

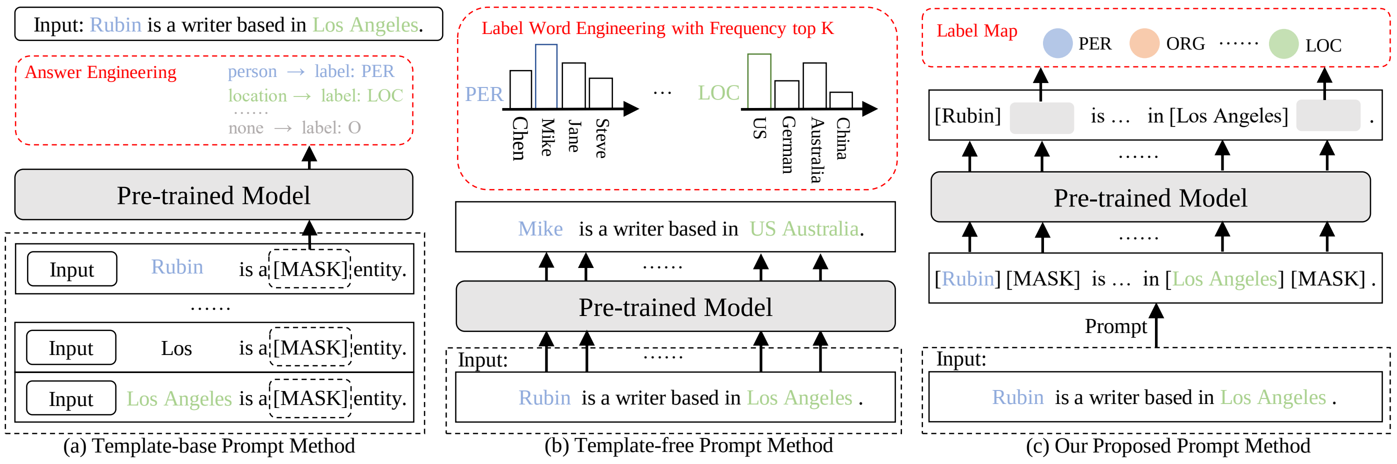
Received: 17 March 2023
Revised: 10 April 2023
Accepted: 26 April 2023
Published: 28 April 2023

1. Introduction

NER is a fundamental task in natural language processing (NLP) that involves locating and identifying named entities in unstructured natural language and categorizing them into a predefined set of classes, such as people, organizations, and locations. Early NER systems were designed based on hand-crafted rules, dictionary matching, and feature engineering, which required domain expertise and a large amount of well-labeled data. As collecting and labeling NER training data is expensive and time-consuming, research on few-shot NER is essential. With the aid of pre-trained language models (PLMs) [1–3], several studies have made significant progress in few-shot NER using techniques such as metric learning [4–6] and label semantics embedding [7,8]. However, discrepancies between the language modeling and NER tasks may limit PLMs' performance. Some researchers are also exploring the application of decomposition networks on few-shot tasks. Decomposing the task into several simple subtasks is common in dealing with complex tasks. However, decomposition networks face the problem of error propagation, and the problem deserves to be considered when we look at supervised learning NER because a large amount of high-quality labeled data can effectively perform joint training of entity location and entity type for NER tasks. When facing a few-shot scenario where a small amount of data is insufficient to train a good network model, decomposing the NER task into several simple tasks is a good solution. Meanwhile, researchers can design the most suitable models and algorithms for different subtasks. There have been several works [9,10] based on decomposition networks with promising performance on few-shot NER tasks.

In recent years, a new training paradigm known as prompt learning has gained traction in NLP tasks, replacing the traditional "pre-train, fine-tune" strategy with a "pre-train, prompt, predict" strategy [11]. Instead of adapting PLMs to downstream tasks, prompt learning reformulates the downstream tasks to better match the pre-trained task of the PLMs. Like ChatGPT [12], a recently popular conversational AI model built on the GPT-3 [13], it also uses the idea of prompt learning, adapting all downstream tasks to generative tasks. Further, suppose the pre-training model is BERT [1], whose pre-training task is the fill-in-the-blank task. Prompt learning adapts the pre-training process by transforming downstream tasks into fill-in-the-blank tasks. To give a concrete example, when recognizing the emotion of a sentence that "I loved this movie.", we may add a prompt template "I felt [MASK]." after the sentence, and ask the model to predict the output at the position of the [MASK] special character. In this case, the model can respond more naturally.

There have been some works demonstrating that prompt learning works well on few-shot NER [14–16]. By bridging the gap between language modeling and the NER task, prompt learning enables PLMs to leverage learned knowledge and achieve better results with fewer training samples. Figure 1a shows a typical prompt-template NER, where all possible entity spans are enumerated and sequentially fed into a pre-trained model for prediction. Similarly, Cui et al. [14] explored a template-based approach using the BART [3], treating NER as a language model ranking problem in a sequence-to-sequence framework. However, these methods suffer from excessive time consumption. To overcome this limitation, Ma et al. [16] proposed a template-free prompt method as shown in Figure 1b. They first obtained label words using a statistical method of a distant dataset and then designed training objectives so that valid entity tokens predicted corresponding label words, while none-entity tokens predicted themselves. This approach avoids enumerating all possible entity spans and solves the time-consuming problem. However, there is still an inconsistency between their training objective and PLMs, leading to inferior results.



Input: Rubin is a writer based in Los Angeles.
Answer Engineering
person → label: PER
location → label: LOC
……
none → label: O
Pre-trained Model
Input Rubin is a [MASK] entity.
……
Input Los is a [MASK] entity.
Input Los Angeles is a [MASK] entity.
(a) Template-base Prompt Method
Label Word Engineering with Frequency top K
PER
Chen
Mike
Jane
Steve
…
LOC
US
German
Australia
China
Mike is a writer based in US Australia.
……
Pre-trained Model
Input:
……
Rubin is a writer based in Los Angeles .
(b) Template-free Prompt Method
Label Map
PER
ORG
……
LOC
[Rubin] is … in [Los Angeles] .
……
Pre-trained Model
……
[Rubin] [MASK] is … in [Los Angeles] [MASK] .
Prompt
Input:
Rubin is a writer based in Los Angeles .
(c) Our Proposed Prompt Method

Figure 1. Different prompt-based NER methods. (**a**) A traditional template-based prompt method [17] that designed prediction templates for all potential entities span. (**b**) A template-free prompt method [16] that maintains the word prediction paradigm of pre-training models to predict a class-related label word at the entity position. (**c**) Our proposed decomposed two-stage prompt learning method that uses a concise prompt template to better fit the pre-training task.

In this work, we propose a decomposed two-stage prompt learning framework in which we decompose the NER task into two sub-tasks, entity locating and entity typing. In Stage I, we collect the distant labels of the dataset and extract only their location information while adopting several strategies to reduce noise introduced by distant labels. In Stage II, we construct a concise but effective prompt template to train the entity typing model, which aligns well with the pre-training task. Figure 1c illustrates the details of the prompt template we designed. With the help of the entity locating model, we can avoid enumerating all potential entity spans when constructing the template, effectively addressing the time-consuming problem. Meanwhile, the entity typing model fully exploits the advantages of prompt learning. In summary, the contributions of this paper can be concluded as follows:

- We propose a few-shot prompt learning framework that decomposes the NER task into two sub-tasks, entity locating and entity typing. The two sub-tasks perform entity localization and type recognition functions separately, addressing the issues of time consumption and inconsistency with pre-trained training objectives.
- We explore the application of distant labels in the entity locating stage accompanied by several experiments. The experimental results show that the entity locating model performed well with the help of distant labels.
- The experimental results demonstrate that in terms of F1 score our framework outperforms other prompt-based methods by 2.3–12.9% on average while the average inference speed is 1313.5 times faster than the template-based method and 1.4 times faster than the enhanced template-free method. These results indicate that our framework strikes a balance between accuracy and inference speed, making it an excellent trade-off for NER tasks.

The remainder of this paper is organized as follows. The related work is introduced in Section 2. Section 4 presents the specific implementation details of our proposed two-stage prompt learning framework. Section 5 introduces our problem setups and describes the datasets and experimental settings. In Section 6, we present the experimental results in detail. Finally, the article is summarized in Section 7.

2. Related Work

2.1. Few-Shot NER

In the past few years, few-shot NER has attracted significant attention from researchers. Fritzler et al. [4] leveraged the prototypical network into the few-shot NER. Hou et al. [7] exploited the matching network to NER. Yang and Katiyar [5] proposed a simple approach that calculated the nearest neighbor of each queried sample as meta-data instead of a prototype, along with a Viterbi decoding method to enhance performance. Huang et al. [6] conducted a comprehensive study of the few-shot NER task, exploring noise-supervised learning and self-learning to improve the results. Ma et al. [8] proposed a simple two-tower NER framework and mined the semantic information of label words. Das et al. [18] introduced a comparative learning scheme that used Gaussian embedding to enhance few-shot NER. Some works have decomposed the NER task into subtasks. For instance, Wang et al. [10] optimized the MRC method for a few-shot setting, and Ma et al. [9] used meta-learning to strengthen decomposed networks. Inspired by this approach, we also proposed a decomposed framework with prompt learning. Specifically, our entity locating model resembles the span detector proposed by Ma et al. [9], but we used the location information of the distant labels to train the entity locating model. Prompt-based few-shot NER methods have also been developed and will be described in Section 2.3. In this work, we follow the few-shot setting of Gao et al. [19], where only a small number of examples are used for training, which is also referred to as the low-resource scenario.

2.2. Distantly-Supervised NER

In the scarcity of high-quality labeled data for NER tasks, distant supervision is an effective solution. Distant supervision uses distant labels generated from knowledge bases or dictionaries to train a model instead of human annotation, but the generated noise labels pose a significant challenge to neural network models. Shang et al. [20] proposed the AutoNER model with the “Tie or Break” scheme and a Fuzzy-LSTM-CRF model to handle tokens with multiple possible labels. Peng et al. [21] formulated the task as a positive-unlabeled learning problem and proposed a novel PU learning algorithm to perform the task using only unlabeled data and named entity dictionaries. Liang et al. [22] proposed a

KB-match method to generate distant labels and explored the application of self-learning algorithms to distantly supervised learning to further improve the model performance. Meng et al. [23] proposed a noise-robust learning scheme for distantly supervised NER, which included a noise-robust loss function and a noisy label removal step. Ying et al. [24] proposed a label refinement method based on contrast learning to improve the accuracy of distant labels. In this work, we also use distant labels in the entity locating stage and use only their location information to reduce part of the noise.

2.3. Prompt Learning NER

Prompt learning methods have recently gained popularity due to the advent of GPT-3 [13] and have demonstrated exceptional performance in a wide range of NLP tasks. Numerous research works have been conducted on prompt learning, with studies focusing on discrete prompt templates [25,26], continuous prompt templates [27–30], and other prompt strategies. Prompt learning's advantage in a wide range of NLP applications has been verified in many kinds of research, including text classification [31–33], relation extraction [33–35], machine translation [36,37], and few-shot learning [19,38–41]. However, there have been only a few studies on NER, especially in the few-shot setting. Cui et al. [14] proposed a template-based prompt learning approach with enumerated entity spans, which makes inference very time-consuming. Chen et al. [15] introduced continuous prompts before each self-attentive layer of the model and then utilized a pointer network to predict the entity. Ma et al. [16] proposed a template-free model that alleviates the cost of enumerating spans but differs from the goal of the pre-training task, which may result in model confusion. In this work, we proposed a two-stage prompt learning framework that decomposes the task into two tasks, entity locating and entity typing. Our prompt learning method is concise because the entity locating model acquires the potential entity span in advance. Furthermore, our entity typing model is designed to better align with the pre-training process of PLMs.

3. Task Definition

3.1. Problem Setup

In this work, we follow the few-shot setting proposed by Gao et al. [19], i.e., only a few annotated examples are available, as in the case of $K = 16$ in that research, which we consider the low-resource scenario of the few-shot setting. Different from traditional domain transfer scenarios, which assume a rich-resource source domain where the size of the source domain is typically much larger than a hundred thousand utterances. Under low-resource scenarios, we cannot learn the knowledge of entity features from a large number of well-labeled datasets of other domains. Thus we do not consider the knowledge transfer problem of new categories between the source and target domains but only consider the low-resource in-domain entity recognition problem. Specifically, when training on a new dataset $D_{new} = \{(X_m, Y_m)\}_{m=1}^{M}$, where X_m is the text input sequence and Y_m is the corresponding sequence of golden labels, we assume only K training examples for each class in the few-shot training set D_{train}. Then, the proposed framework is tested with an unseen test set D_{test}.

Here, for the few-shot NER task, we decompose it into two sub-tasks, entity locating and entity typing. For the entity locating stage, we explore the application of the in-domain distant labels $\{(X_m, D_m)\}_{m=1}^{M}$. Obviously, the entity locating model should be type-agnostic, i.e., we do not differentiate the specific entity type. In such a setting, we re-label the in-domain distant labels and few-shot training set with the BIOES tagging scheme following the study of Ma et al. [9], i.e., B-ORG, I-ORG, E-ORG, S-PER, B-LOC, E-LOC, which provides more specific and fine-grained boundary information of entity spans. Additionally, we only retain the label location information, i.e., ignore the class type such as ORG, PER, and LOC, and maintain the label prefix information B, I, O, E, and S. Here, S denotes a single entity; B, I, and E denote the start, inner, and end positions of an entity, respectively; and O denotes a non-entity. The new distant labels $\{(X_m, D'_m)\}_{m=1}^{M}$ and new few-shot training set D'_{train} will be used to train the entity locating model. Only the few-shot training set D_{train} is used to train the entity typing model for the entity typing stage.

In the evaluation setup, we use a standard evaluation setup that considers the entire development dataset D_{dev} and test dataset D_{test}.

3.2. Distant Label Construction

For distant labels, we use the distant labels provided by Liang et al. [22]. Given input $X = \{x_i\}_{i=1}^{n}$ with n tokens, we use their KB-matching method to obtain the distant labels $D = \{d_i\}_{i=1}^{n}$. The KB-matching method will automatically generate the distant labels set $\{(X_m, D_m)\}_{m=1}^{M}$ for unannotated data in the dataset by querying ultra-large open-source databases such as the Wiki database. Specific technical details can be found in Appendix A.

3.3. Few-Shot Data Construction

Unlike sentence-level classification tasks, token-level classification tasks such as NER tasks may contain multiple entity classes in one input, which leads to uneven distribution of entity categories in a few-shot dataset. Thus, we cannot simply extract K training samples as a K-shot dataset. Given the dataset $\{(X_m, Y_m)\}_{m=1}^{M}$, we conduct the greedy sampling strategy [5] to build the few-shot training set D_{train}. The greedy sampling strategy samples the dataset according to entity frequencies in order to avoid uneven distribution of entity categories. Specific technical details can be found in Appendix B.

4. Model

This work proposes a decomposed two-stage prompt learning framework for few-shot named entity recognition. As shown in Figure 2, the overall framework comprises two stages: entity locating and entity typing. Along the arrow direction, we show the complete inference flow when dealing with the NER task. In the figure, we take a simple example to illustrate this processing. Given a text input, “Rubin is a writer based in Los Angeles”. In Stage I, we use the entity locating model to locate entities in the input. Specifically, we use the entity locating model to predict the input text sequence as a label sequence containing only BIOES information. Based on the BIOES information, we can extract the two possible entities in the input: Rubin and Los Angeles. Then in Stage II, we design a concise prompt template and reconstruct the template input based on those extracted entities. We use the entity typing model to predict the template input and finally obtain the result that Rubin is a person entity and Los Angeles is a location entity. In order to better describe the two-stage prompt learning framework, we will use different sections to introduce the specific details of the model in the two stages separately. In Section 4.1, we present the entity locating model and its training process. We then detail the entity typing model and its training process in Section 4.2. Finally, we introduce the overall inference process of the proposed framework in Section 4.3, which solves the NER task in an elegant pipeline approach.

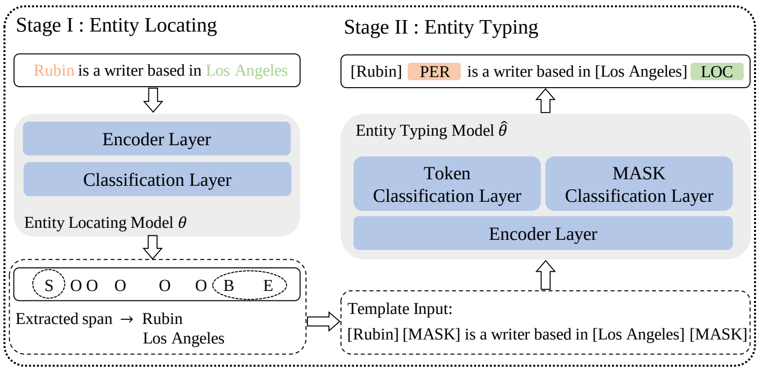

Stage I : Entity Locating
Rubin is a writer based in Los Angeles
Encoder Layer
Classification Layer
Entity Locating Model θ
S O O O O O B E
Extracted span → Rubin
Los Angeles
Stage II : Entity Typing
[Rubin] PER is a writer based in [Los Angeles] LOC
Entity Typing Model θ̂
Token Classification Layer
MASK Classification Layer
Encoder Layer
Template Input:
[Rubin] [MASK] is a writer based in [Los Angeles] [MASK]

Figure 2. Overall view of our decomposed two-stage prompt framework. Different colors mean different entities. In this example, orange means person entity, and green means location entity.

4.1. Stage I: Entity Locating

In this section, we focus on the entity locating model structure and the formulas involved in Section 4.1.1, and introduce the complete training process of the entity locating model in Section 4.1.2.

4.1.1. Entity Locating Model

We used a simple sequence labeling method to implement the entity locating model. The structure of the entity locating model consists of two layers, the encoder layer and the classifier layer. Given an input sequence pair (x, y), we first initialize a pre-training model f_θ as entity locating encoder. For the text input sequences $x = \{x_1, x_2, \cdots, x_n\}$ with n tokens, we pass them through the encoder layer to obtain the contextualized representations $h = \{h_1, h_2, \cdots, h_n\}$ as follows:

$$h = f_\theta(x) \tag{1}$$

We then feed all hidden vectors h generated by f_θ into a linear classification layer with a softmax function to compute the probability distribution of labels for each h_i:

$$p(x_i) = softmax(Wh_i + b) \tag{2}$$

where $p(x_i) \in \mathbb{R}^{|C|}$, C is the label search space containing $\{B, I, O, E, S\}$, and $\{W, b\}$ are randomly initialized trainable parameters. Following Wu et al. [42], the training loss is modeled as the averaged cross-entropy of the predicted label distribution and the distant label over all tokens plus a maximum term of the above loss, as:

$$L(\theta') = \frac{1}{n}\sum_{1}^{n} CrossEntropy(y_i, p(x_i)) + \beta \max_{i\in\{1,2\cdots,n\}} CrossEntropy(y_i, p(x_i)) \tag{3}$$

where $\beta \geq 0$ is a weighting factor.

4.1.2. Training of Entity Locating Model

To obtain accurate location information in the target domain, we explored the application of distant labels in the entity locating model. The new distant dataset $\{(X_m, D'_m)\}_{m=1}^{M}$ are given for training the entity locating model. However, the distant labels still provide a large amount of noise, so we need to alleviate the noise problem further.

One crucial strategy we used in the training process is early stopping. Since our entity locating model is susceptible to overfitting the noise, the early stopping strategy acts as a powerful regularization technique to prevent overfitting and helps the model maintain its generalization capabilities. We update the model by the loss function mentioned in Equation (3). Upon reaching an excellent early stopping step T, we terminate the training in advance and get an excellent noisy model θ^T.

Inspired by Jiang et al. [43], in addition to the early stopping strategy, we propose to fine-tune the noise model θ^T with strongly labeled few-shot dataset D_{train}. We then select the model that performs best on the development dataset D_{dev} to be the final saved entity locating model θ_n^T. The fine-tuning strategy is a highly effective method that enables the model to acquire the correct in-domain knowledge from the few-shot data, thereby allowing it to better adapt to the target domain and improve its performance.

The specific training details are available in Algorithm 1.

Algorithm 1 Pseudo-code of the training for entity locating.

```
Input: distant dataset $\{(X_m, D'_m)\}_{m=1}^{M}$ ; The pretrained model $f_\theta$; The early stopping time
  $T$; The updating formula of Adam $\sigma$.
  for $t = 1, 2, \cdots, T$ do
    Sample a minibatch $B$ form $\{(X_m, D'_m)\}_{m=1}^{M}$
    Calculate the loss $L(\theta)$ according to Equation (3)
    Update the model using Adam $\sigma$:
            $\theta^t = \sigma(\theta^{(t-1)}, B)$
  end for
Output: The early stopped model $\theta^T$
  // Fine-tune with few-shot data
Input: The few-shot data $D_{train}, D_{dev}$; The early stopped model $\theta^T$; The Epoch $N$.
  for $t = 1, 2, \cdots, N$ do
    Sample a minibatch $B$ form $D_{train}$
    Fine-tune $\theta^T$ with $B$
    if $Eval(\theta^T)$ is best then
      Save model $\theta_n^T$
    end if
  end for
Output: The best entity locating model $\theta_n^T$
```

4.2. Stage II: Entity Typing

In this section, we introduce the proposed prompt template and the structure of the entity typing model in Section 4.2.1 and its training process in Section 4.2.2.

4.2.1. Entity Typing Model

Before introducing the entity typing model, we first explain the construction of the prompt template. Figure 2 (Stage II) shows a clear case:

Rubin is a writer based in Los Angeles.

Where the correct entity span is Rubin and Los Angeles. We insert brackets around the correct span and follow it with [MASK] special token as below:

[Rubin] [MASK] is a writer based in [Los Angeles] [MASK].

The advantage of the proposed prompt template is that it is concise enough to understand, yet well adapted to the pre-training task, which helps the entity typing model fully utilize the knowledge of the PLMs in the case of a few-shot setting.

For convenience, we mark $\mathcal{T}$ as the prompt process. Given an input sequence $x_{in} = \{x_1, x_2, \cdots, x_m\}$ with m tokens, we first transform the input x_{in} into the template form x', where $x' = \mathcal{T}(x_{in})$. The golden labels y' can be written as $y' = \{y_{mask_1}, \cdots, y_{mask_l}\}$ which mainly indicates the entity type of the [MASK] token, where l is the number of the entity.

The structure of the entity typing model contains three layers: the encoder layer, mask classification layer, and token classification layer. We re-initialize a pre-training model $g_{\hat{\theta}}$ as an entity typing encoder. Next, we use the encoder $g_{\hat{\theta}}$ to compute contextual token representations $h' = \{h'_i\}_{i=1}^{M}$ in the same way as Equation (1):

$$h' = g_{\hat{\theta}}(x') \tag{4}$$

where M is the length of the template form x'.

After h' has passed through the mask classification layer, we can calculate the probability distribution of different entity types at the [MASK] token using Equation (5):

$$\begin{aligned} q(x'_{[MASK]}) &= q\big([MASK] \mid x'\big) \\ &= softmax\Big(w \cdot h_{[MASK]} + b\Big) \end{aligned} \tag{5}$$

where the $q\left(x'_{[MASK]}\right) \in \mathbb{R}^{\left|C'\right|}$, C' is the entity type set.

We then use the CrossEntropy function to calculate the loss between the golden label and the predicted label probability distribution, as:

$$L_{type} = CrossEntropy\left(y', q(x'_{[MASK]})\right) \tag{6}$$

It should be noted that the mask classification layer only takes into account the position of the [MASK] token and ignores the other tokens in the input sequence. However, these other tokens also play a role in model training. To give equal attention to all tokens, we propose a token classification layer based on the pre-training objective of the masked language model. This layer allows the remaining tokens to predict themselves, enhancing the training process. The training objective can be considered as maximizing the probability $P(x'_i \neq [MASK] \mid x')$ of the x' except [MASK] token, as below:

$$L_{token} = -\sum_{i=1}^{M} \log P\left(q(x'_i) = x'_i | x'_i \neq [MASK]\right) \tag{7}$$

where $q(x'_i) = softmax(FFN(h'_i))$. Similar to Equation (2), we feed all hidden vectors h' into an FFN (feed-forward network) with a softmax function to compute the probability. The added token classification layer increases the difficulty of model training, thus finally improving the robustness and stability of the model.

The final loss of entity typing model combines the above two losses and can be written as:

$$\begin{aligned} L_{finall} &= L_{type} + L_{token} \\ &= CrossEntropy\left(y', q\left(x'_{[MASK]}\right)\right) - \sum_{i \neq [MASK]}^{M} \log P\left(x'_i \neq [MASK] \mid x'\right) \end{aligned} \tag{8}$$

4.2.2. Training of Entity Typing Model

We start by reconstructing the few-shot dataset D_{train} into the prompt template dataset $D^*_{train} = \mathcal{T}(D_{train})$. It should be noted that the input sentences in the few-shot dataset D_{train} contain at least one named entity, while each prompt template input in D^*_{train} contains at least one [MASK] token, enabling the correct calculation of the loss L_{token}.z'

We then fine-tune the entity typing model $\hat{\theta}$ using the designed loss written in Equation (8), and select the best model that performs in the development dataset D_{dev} as the final saved entity typing model $\hat{\theta}_n$. The complete training process is presented in Algorithm 2.

Algorithm 2 Pseudo-code of the training for entity typing.

```
Input: The few-shot data $D_{train}, D_{dev}$; The pretrained model $g_{\hat{\theta}}$; The Epoch N;
  Prompt process $\mathcal{T}$.
  construct the prompt template dataset $D^*_{train} = \mathcal{T}(D_{train})$
  for $t = 1, 2, \cdots, N$ do
    Sample a minibatch $B$ form $D^*_{train}$
    Calculate the loss $L(\hat{\theta})$ according to Equation (8)
    if $Eval(\hat{\theta})$ is best then
      Save model $\hat{\theta}_n$
    end if
    Update the model using Adam $\sigma$
  end for
Output: The best entity typing model $\hat{\theta}_n$
```

4.3. Inference Process

In the inference step, we can easily handle the entire NER task using a pipeline. Now, given a test dataset D_{test}, we use the saved entity locating model θ_n^T to predict all possible entity spans in the input sentence. Then, we generate new template input through the prompt process $\mathcal{T}$ and finally feed them into the saved entity typing model $\hat{\theta}_n$ to predict the type of those entity spans.

To more intuitively describe the pipeline approach in inference. We take input from the test dataset D_{test} as an example, as follows:

The construction of Hong Kong Disneyland began two years ago in 2003.

The input sequences are mapped one by one into label sequences through the well-trained entity locating model, as below:

O O O B I E O O O O O O O

The position tag in the label sequence will indicate all possible entities in the input sequence. "Hong" is mapped to "B" through the entity locating model, "Kong" is mapped to "I", and "Disneyland" is mapped to "E" in the same way. B, I, and E indicate that "Hong Kong Disneyland" are the start, inner, and end positions of an entity, respectively. Thereby, these three tokens are treated as one complete entity.

Then, the extracted candidate entity "Hong Kong Disneyland" is used to help build the new prompt template input as:

> The construction of [Hong Kong Disneyland] [MASK] began two years ago in 2003.

Finally, the well-trained entity typing model predicts the [MASK] token of the prompt template input as the facility type. Therefore, "Hong Kong Disneyland" is finally recognized as a facility entity.

Our proposed framework has the advantage of outputting the result at once for each query, thus alleviating the time-consuming problem of prompt learning on NER tasks. By using this pipeline approach, we can efficiently and accurately extract named entities from the input text.

5. Experiment Settings

5.1. Implementation Details

For a fair comparison, we keep the same hyperparameters as below for all other comparison methods. We conduct experiments with $K \in \{5, 10, 20, 50\}$. We use three different training sets for each experiment and set four random seeds to calculate the average F1 result. We use the "bert-base-cased" pre-trained model from Huggingface Library [44] as the base encoder model. The hyperparameter settings are shown below: learning rate = 10^{-4}, batch size = 4, and epoch = 20 for the few-shot setting. As for the optimizers, we use AdamW [45] with a 2% linearly scheduled warmup. As for distant labels training, we use grid search to get the best hyperparameter, where the search space of learning rate is $\{10^{-5}, 5 \times 10^{-5}, 10^{-4}\}$, the number of mini-batch is selected from {16, 32}, and the β is set to 2.

For the early stopping strategy, we follow the setting of Liang et al. [22], which uses the training step as the unit to evaluate the model. Specifically, we evaluate the performance of the entity locating model every 100 steps. Their research shows that using the step to evaluate the model allows it to stop training earlier than using epoch as the unit, preventing the model from overfitting to the noise and allowing it to perform better. In practice, we set 500 steps (approximately one epoch) for the CoNLL2003 dataset and 1500 steps (approximately two epochs) for the OntoNote 5.0 dataset.

In the hardware setup, our server is equipped with a single NVIDIA RTX 3090 GPU with 24 GB of GPU memory, which has 10,496 CUDA cores. The CUDA version we are using is 11.4. For software libraries, we list the following important libraries and indicate the versions used in our experiments: Transformer version 4.9.2, torch version 1.8.1, numpy version 1.2.14, and seqeval version 1.2.2. The Python version is 3.8.16.

5.2. Datasets

We evaluate our framework on the two benchmark datasets from different domains, CoNLL2003 (news) [46] and OntoNotes5.0 (general domain) [47].

- The CoNLL2003 is a common open-source dataset for the news domain. It includes four types of entities (i.e., PER, LOC, ORG, and MISC).
- The OntoNote5.0 dataset includes 18 types of entities. Our model focuses on 11 named entities, i.e., PER, NORP, FAC, ORG, GPE, LOC, PRODUCT, EVENT, WORK OF ART, LAW, and LANGUAGE. Similar to Ma et al. [16], we exclude seven entity types related to value, numerical, time, and date (i.e., DATA, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, and CARDINAL) in our training and evaluation. These excluded entity types are marked as non-entities in our approach.

Table 1 shows the details of the datasets, including the domain, the number of types, and the size of the train and test dataset.

Table 1. Dataset details.

Datasets	Domain	Type Number	Training Set Size	Test Set Size
CoNLL2003	News	4	14.0 k	3.5 k
OntoNotes5.0 *	General	11	60.0 k	8.3 k

* means the dataset omits value/number/time/date entity types.

Table 2 shows the results of the few-shot dataset size to the original dataset for different values of *K*. The size of few-shot samples accounts for less than 1% of the size of the whole dataset, demonstrating that our experiments are conducted in a low-resource scenario.

Table 2. Percentage of few-shot dataset size to the whole dataset size at different values of *K*.

Datasets	$K = 5$	$K = 10$	$K = 20$	$K = 50$
CoNLL2003	0.07%	0.14%	0.26%	0.59%
OntoNotes5.0	0.06%	0.13%	0.27%	0.64%

For a fair comparison, we directly used the few-shot dataset provided by Ma et al. [16]. Our model and the comparison models are trained and evaluated on the same few-shot training and test sets.

5.3. Evaluation Index

In this paper, we take the weighted summed average F1 as the evaluation index. Before introducing the F1 equation, we introduce the concepts of precision and recall as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{10}$$

where TP is the number of entities correctly predicted by the model, FP is the number of incorrectly predicted entities, and FN is the number of entities that are false negatives. TP + FP represents the number of all predicted entities and TP + FN is the number of gold entities.

Precision measures how often the model is correct when it predicts a positive instance. Recall measures how well the model is able to find all the positive instances in a dataset. F1 scores represent the harmonic mean of precision and recall, it can be calculated as:

$$\text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{11}$$

In this work, we use span F1 and typing F1 to evaluate the performance of the entity locating model and entity typing model, respectively. This is the same as Equation (11), where span F1 is calculated considering only the entity span while typing F1 considers the entity type when the entity span is 100% determined. In the inference, the final F1 value needs to consider both entity span and entity type.

6. Results

6.1. Baselines

We compare our method with the following baselines. All pre-trained models used in the baseline methods are "bert-base-cased", except Template NER, whose based model is BART [3]:

- BERT-NER: The BERT-based model provides a strong baseline that fine-tunes the BERT model with a label classifier. For NER, the hidden vectors of each token are fed into the classifier to calculate the probability distribution.
- NNShot and StructShot [5]: The classical metric-based few-shot approaches. NNShot leverages a nearest-neighbor classifier for few-shot prediction after pre-training on the source domain. StructShot further proposed a method to capture the label dependencies between entity tags by using the abstract tag transition distribution estimated on the source domain data. Note that these two approaches are based on domain transfer settings. Following Ma et al. [16], we also considered the scenario where the source domain is unavailable. For StructShot, we use distant labels to provide the tag transition distribution rather than the source domain data from the original paper.
- TemplateNER [14]: The method enumerates all possible entities span matching with all the entity types and then generates the prompt templates for training. In inference, it queries all the templates to the pre-trained model to calculate the scores and finds the most probable entity by score ranking.
- EntLM and EntLM-Struct [16]: The template-free prompt method uses distant labels to obtain prototype words of each type in the dataset. Then, it designs a task in which the target entity token goes through the pre-trained model to predict the prototype words of the corresponding type while the O-tag token predicts itself. As for EntLM-Struct, they use the same method as StructShot to further extend EntLM.

6.2. Main Results

We take F1-score as the evaluation index, the results of our proposed method and baselines under the few-shot setting are shown in Table 3.

We observe: (1) Our method performs consistently better than the other baseline on all the datasets under all the K-shot settings. On the CoNLL2003 dataset, the results of our method slightly surpass the one of EntLM-Struct except for 10-shot. Additionally, the stability of our proposed model is slightly worse than EntLM. On the OntoNote5.0 dataset, the F1 score of our proposed model far exceeds those of other methods. Specifically, our method outperforms the EntLM-Struct by up to 9.2% on average and surpasses the Template NER by + 16.5% on average. Additionally, those two methods are the main ones we compare in prompt learning. (2) Without a rich-resource domain, the metric-based methods NNShot do not show advantages over the other methods, which shows the limitation of these methods under more practical few-shot scenarios.

To evaluate a model, we need to consider not only its performance but also its efficiency to avoid suffering from the time-consuming problem of template-base prompt learning mentioned in Section 1. In this work, we also performed experiments on the efficiency of inference, as demonstrated in Table 4.

Table 3. Main results of our framework and compared baselines on two datasets under different few-shot settings (K = 5, 10, 20, 50).

Datasets	**Methods**	K **= 5**	K **= 10**	K **= 20**	K **= 50**
CoNLL2003	BERT-NER	41.87 (±12.12)	59.91 (±10.65)	68.66 (±5.13)	73.20 (±3.09)
	NNShot †	42.31 (±8.92)	59.24 (±11.71)	66.89 (±6.09)	72.63 (±3.42)
	StructShot †	45.82 (±10.30)	62.37 (±10.96)	69.51 (±6.46)	74.73 (±3.06)
	Template NER †	43.04 (±6.15)	57.86 (±5.68)	66.38 (±6.09)	72.71 (±2.13)
	EntLM	53.43 (±2.26)	63.32 (±2.78)	69.11 (±2.01)	73.58 (±1.50)
	EntLM+Struct	53.10 (±7.67)	64.79 (±3.85)	69.51 (±2.60)	73.60 (±1.56)
	Our	**55.38 (±6.41)**	**66.75 (±5.22)**	**70.15 (±3.46)**	**75.07 (±1.37)**
OntoNote5.0	BERT-NER	34.77 (±7.16)	54.47 (±8.31)	60.21 (±3.89)	68.37 (±1.72)
	NNShot †	34.52 (±7.85)	55.57 (±9.20)	59.59 (±4.20)	68.27 (±1.54)
	StructShot †	36.46 (±8.54)	57.15 (±5.84)	62.22 (±5.10)	68.31 (±5.72)
	Template NER †	40.52 (±8.86)	49.89 (±3.66)	59.53 (±2.25)	65.15 (±2.95)
	EntLM	44.80 (±3.68)	54.67 (±3.04)	66.00 (±1.82)	72.54 (±1.07)
	EntLM + Struct	45.62 (±10.26)	57.30 (±4.18)	67.90 (±3.50)	73.57 (±2.10)
	Our	**60.81 (±4.64)**	**69.50 (±4.36)**	**74.22 (±2.29)**	**76.80 (±1.23)**

† denotes the results reported in Ma et al. [16]. Bold values denote the best results.

Table 4. The inference time (ms) of each method on all the test datasets.

Method	**CoNLL**	**OntoNotes**
BERT-NER †	19.59	23.03
TemplateNER †	14,836.57	48,425.06
EntLM	20.64	24.48
EntLM + Struct	29.71	33.75
Our	21.03	25.23

† denotes the results reported in Ma et al. [16].

We take BERT-NER as a baseline and observe that: (1) Our method achieves comparable speed with BERT-NER. With the help of the entity locating model, our method identifies the types of all potential entities by inference only once. (2) Compared with the other two prompt learning methods, we are 707.8 times faster than TemplateNER and 1.4 times faster than EntLM-Struct on the CoNLL. Additionally, we are 1919.1 times faster than TemplateNER and 1.3 times faster than EntLM-Struct on the OntoNotes. (3) We can draw essential conclusions in Figure 3, where our model achieves the best trade-off between accuracy and speed compared to other methods.

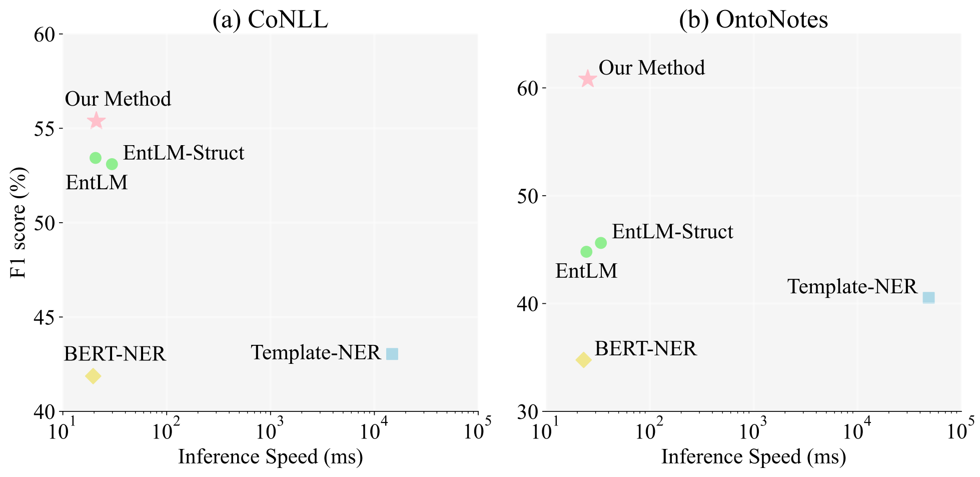


Figure 3. Inference accuracy vs. speed for different models evaluated on a 5-shot setting. Our method achieves the best trade-off between accuracy and speed compared to other methods.

6.3. Ablation Study

In this section, we evaluate ablation experiments on entity locating and entity typing models of our framework.

6.3.1. Ablation Experiments of Entity Locating

To validate the effect of different denoising strategies, we introduce the following variants and baselines for the ablation study: (1) w/o distant labels, where we fine-tune the entity locating model only with few-shot data. Therefore, there is also no early stopping strategy. (2) w/o fine-tune means we only train the model with distant labels with an early stopping strategy and do not introduce few-shot in-domain data. (3) w/o early stopping, for this setting, we explore the impact of the early stopping strategy. (4) w/o early stopping and w/o fine-tune, under this setting, we explore the results obtained by training the model with only distant labels.

Table 5 highlights the contributions of each method in our entity locating model. Generally speaking, removing any of them will lead to a performance drop. We can conclude the following: (1) When we drop the use of distant labels, the model's performance drops dramatically, which means distant labels provide powerful in-domain location information. (2) Fine-tuning with few-shot data and an early stopping strategy still have a positive effect. We further research and show the results in Figures 4 and 5. (3) On OntoNotes, the entity locating model achieves good accuracy, which is probably the reason why the final score is much higher than other methods.

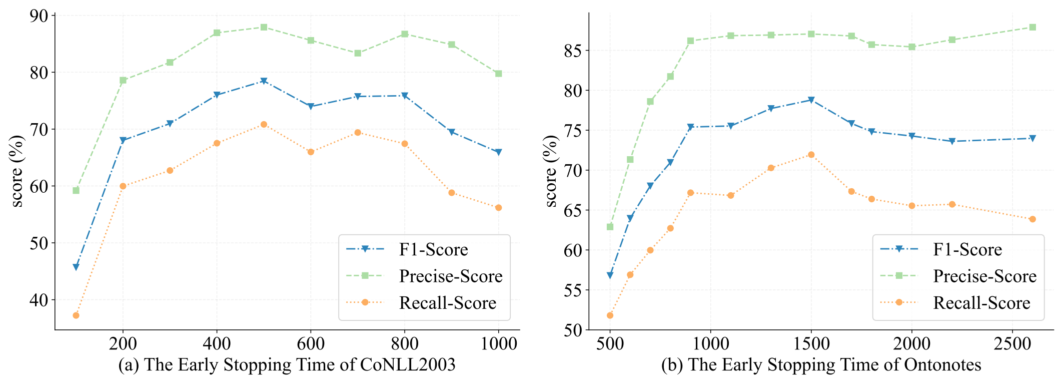


Figure 4. The effect of early stopping time on the performance of entity locating models.

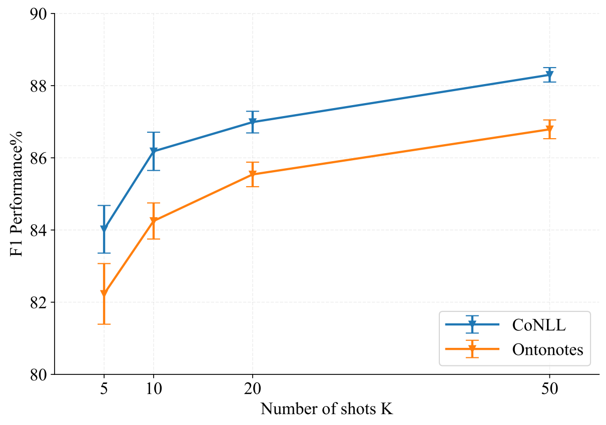


Figure 5. Performance comparison under two benchmark datasets with different k-shot data.

Table 5. Ablation study for entity locating model: span average F1 scores on 5-shot.

Model	CoNLL (Span F1)	OntoNotes (Span F1)
Stage I (entity locating)	84.02	84.23
w/o distant labels	55.45	74.14
w/o fine-tune	78.45	81.80
w/o early stopping	80.22	83.23
w/o early stopping and w/o fine-tune	71.42	80.63

To explain the excellent performance on the OntoNote5.0 dataset, we performed further analysis under the five-shot, as shown in Table 6. We measure the performance of the entity locating model and entity typing model in terms of span F1 and type F1. Obviously, for the comparison method, EntLM-Struct does not decompose the NER task; therefore, span F1 and type F1 are shown as N/A. We set three different baselines for the proposed framework: (1) the original scheme, the best framework design scheme we proposed; (2) w/o distant labels in Stage I, where only few-shot datasets are used for training the entity locating model in Stage I; (3) w/o fine-tune in Stage I, where only distant labels are used for training the entity locating model in Stage I. For Stage II, we keep the same settings in all three baselines, so the type F1 score in the table remains unchanged.

Table 6. Average F1 scores of the framework at different stages on the OntoNotes under 5-shot.

Model	Span F1	Type F1	Finall F1
EntLM-Struct	N/A	N/A	45.62
Our	84.23		60.81
w/o distant labels in Stage I	74.14	70.94	49.26
w/o fine-tune in Stage I	81.80		54.38

From the results, we can make the following analysis: (1) Even with only few-shot data for training in the entity locating stage, our final average F1 score is still about 3% higher than the average F1 score of the comparison method in Table 3. It indicates that our framework design itself has an advantage in the OntoNote 5.0 dataset. Additionally, the model performance is further improved with the help of the distant labels. (2) Distant labels provide more information on entity locations due to their rich in-domain location information, while few-shot datasets are much smaller than the original dataset, which accounts for only about 0.07% in five-shot. Therefore, the information provided by the few-shot dataset may be biased, and using them to train the model may damage the model's performance. (3) The final F1 score is almost the same as the cumulative multiplication result of span F1 and type F1 score, which illustrates the error propagation problem in the decomposition network. Especially in the scenario where training samples are scarce, the error propagation problem is inevitable. Our framework employs some strategies mentioned in Section 4.1.2 to mitigate this problem in the entity locating stage, which leads to an 11.5% improvement in the final average F1.

We performed further research on the early stopping strategy, and the results of the study are shown in Figure 4. Three curves represent the precision, F1, and recall score change, respectively. Figure 4a illustrates the variation curve of the performance under the CoNLL2003 dataset. The best results are achieved when the early stopping time reaches 500, and all three curves showed a downward trend as the steps increased. As shown in Figure 4b, under the OntoNote5 dataset, we obtain the best results at 1500 steps. When the number of training steps increases, the precision score curve shows an increasing trend while the recall score gradually decreases, resulting in a decreasing trend of the F1 score curve, which is a phenomenon of the model overfitting to the distant labels.

To evaluate the impact of different few-shot dataset sizes on the model's performance, we conducted 12 experiments with three sampled few-shot datasets and four random seeds, with K set to $\{5, 10, 20, 50\}$. Figure 5 displays the line graph of the model's performance at different values of K for the two benchmark datasets. The graph analysis suggests that: (1) The model's performance tends to increase as the few-shot dataset grows, with the fastest growth at $K = 10$, after which the performance improvement slows down. In terms of data cost, the 10-shot training set provides the greatest benefit to the entity locating model. (2) The standard deviation of the results indicates the degree of fluctuation in the experimental results, with a smaller fluctuation indicating a more stable model. (3) The above observations are consistent for both benchmark datasets, demonstrating the positive role of our fine-tuning strategy in improving model performance and stability.

In this section, we conduct a series of ablation experiments on our entity locating model. To summarize: (1) the distant labels provide the entity locating model with powerful in-domain entity localization information; (2) with the help of the early stopping strategy and the fine-tuning strategy, the model can avoid overfitting the noise of distant labels.

6.3.2. Ablation Experiments of Entity Typing

In this section, we design ablation experiments for the proposed entity typing model. In Figure 6, we designed three experiments as follows: (1) *w/o brackets*, we remove the brackets used to "highlight" the entities to explore the impact of these brackets on model performance; (2) *not [MASK]*, to explore the importance of the [MASK] special character in the prompt learning, we replace [MASK] with a textual character, such as "type"; (3) *w/o token loss*, we remove the token loss mentioned in Equation (7), which indicates that we will ignore a portion of the samples that do not contain any entities which we can consider as negative samples, and we design this ablation experiment to verify the effect of negative samples on the model performance.

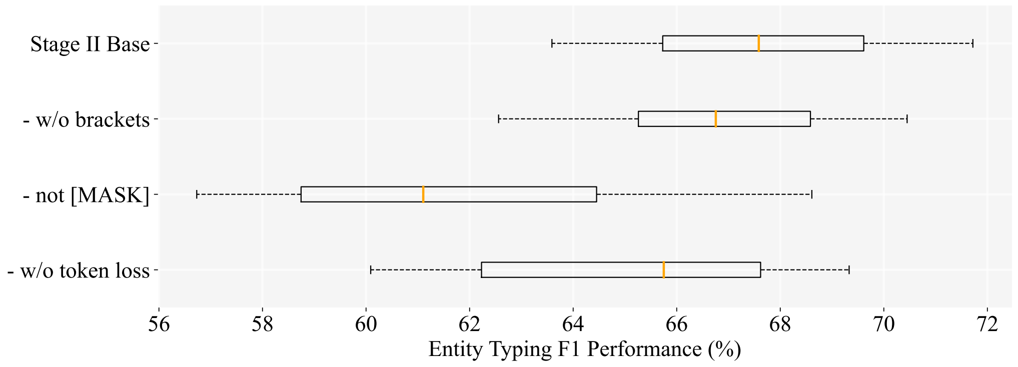


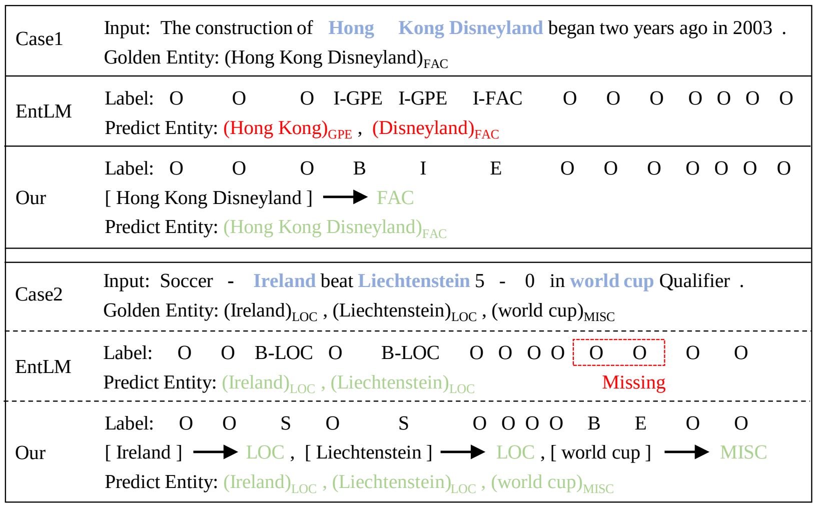
Figure 6. Ablation experiment of entity typing model on 5-shot under CoNLL2003.

We similarly sampled three five-shot data and four random seeds for a total of 12 experiments to draw the boxplot. We can make the following analysis from Figure 6: (1) The model's overall performance decreases after we remove the brackets. It suggests that the brackets help the model to "highlight" the entities which need to be identified. (2) When we replace [MASK] special characters with other textual characters, the model performance drops significantly: the median F1 value drops by 6%, and the highest F1 score is also lower than the F1 score of other settings. In addition, the box width is longer, indicating a more unstable model performance. (3) When we remove the token loss, the model performance decreases, and the model stability worsens, indicating that negative samples play an active role in the model training.

To conclude, we propose a prompt-based entity typing model, and the results of the ablation experiments demonstrate the effectiveness of our designed template and loss function.

6.3.3. Case Study

Additionally, we give two typical case studies, depicted in Figure 7. In case 1, we observed that EntLM identifies the complete entity as two different entities. This happens because EntLM still employs the token classification strategy and disregards the completeness of the entity, while our framework successfully identifies the whole entity span and classifies it correctly. In case 2, EntLM fails to detect an entity because it appeared during the training process as "World Cup" instead of "world cup". However, our entity locating model has better generalization capabilities.



Case1	Input: The construction of Hong Kong Disneyland began two years ago in 2003 . Golden Entity: (Hong Kong Disneyland)$_{FAC}$
EntLM	Label: O O O I-GPE I-GPE I-FAC O O O O O O O Predict Entity: (Hong Kong)$_{GPE}$, (Disneyland)$_{FAC}$
Our	Label: O O O B I E O O O O O O O [Hong Kong Disneyland] $\longrightarrow$ FAC Predict Entity: (Hong Kong Disneyland)$_{FAC}$
Case2	Input: Soccer - Ireland beat Liechtenstein 5 - 0 in world cup Qualifier . Golden Entity: (Ireland)$_{LOC}$, (Liechtenstein)$_{LOC}$, (world cup)$_{MISC}$
EntLM	Label: O O B-LOC O B-LOC O O O O O O O O Predict Entity: (Ireland)$_{LOC}$, (Liechtenstein)$_{LOC}$ Missing
Our	Label: O O S O S O O O O B E O O [Ireland] $\longrightarrow$ LOC , [Liechtenstein] $\longrightarrow$ LOC , [world cup] $\longrightarrow$ MISC Predict Entity: (Ireland)$_{LOC}$, (Liechtenstein)$_{LOC}$, (world cup)$_{MISC}$

Figure 7. Case study: explore the performance of EntLM and our proposed two-stage framework. Note that the blue font denotes the gold entities in sentences, the red font indicates a wrong prediction, and the green font indicates a correct prediction.

7. Conclusions

In this paper, we propose a novel decomposed two-stage prompt learning framework for tackling NER tasks using prompt learning. The proposed framework comprises an entity locating model that predicts the entity span and an entity typing model that uses prompt learning to recognize the type of predicted span using a concise yet effective prompt template. Our approach offers a promising solution to address the time-consuming problem associated with NER tasks, and the experimental results demonstrate its effectiveness. In future research, we plan to focus on developing a more general entity locating model by leveraging extensive open-source noise data. We hope that our work will inspire further research and development in other natural language processing tasks beyond NER, by adapting the prompt template and fine-tuning the model on a task-specific dataset.

Author Contributions: Conceptualization, F.Y. and L.H.; methodology, F.Y.; software, F.Y.; validation, F.Y. and S.L.; formal analysis, F.Y. and S.L.; investigation, F.Y. and S.L.; resources, L.H. and K.C.; data curation, F.Y.; writing—original draft preparation, F.Y.; writing—review and editing, L.H. and K.C.; visualization, S.L.; supervision, L.H.; project administration, L.H. and K.C.; funding acquisition, L.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the Fundamental Research Funds for the Provincial Universities of Zhejiang under Grant RF-B2022002 and the National Natural Science Foundation of China under Grant 62072410.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. The distant labels can be accessed at https://github.com/cliang1453/BOND/tree/master/dataset, (accessed on 1 November 2022) and the few-shot data can be accessed at https://github.com/rtmaww/EntLM/tree/main/dataset, (accessed on 1 September 2022).

Conflicts of Interest: The authors declare no conflict of interest.

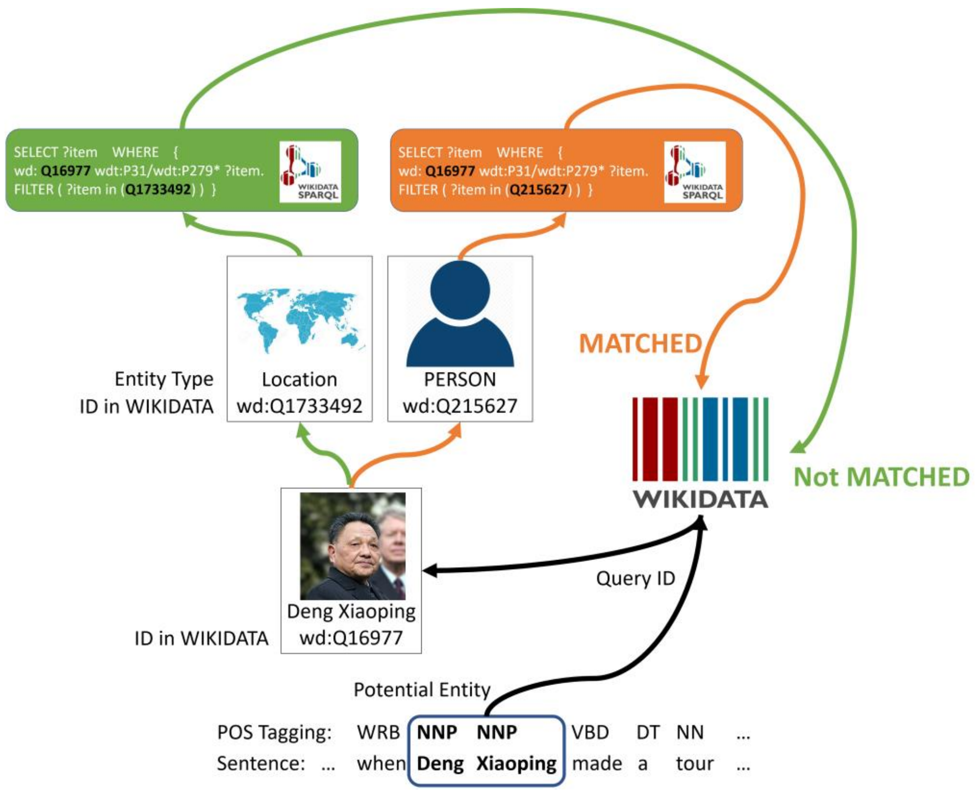
Abbreviations

The following abbreviations are used in this manuscript:

NER	Named Entity Recognition
NLP	Natural Language Processing
PLM	Pre-Trained Language Model
MRC	Machine Reading Comprehension
FFN	Feed-Forward Network
BERT	Bidirectional Encoder Representation from Transformers
GPT-3	Generative Pre-Trained Transformer 3

Appendix A. KB-Matching Method

The KB-Match method [22] follows a two-step approach for entity typing. Firstly, it identifies potential entities using POS tagging and queries the parent categories of an entity in the knowledge tree using SPAQL. The querying process continues until a category corresponding to a type is found. The method then builds a multisource gazetteer by crawling online data sources and matches an entity with a type if it appears in the gazetteer for that type. Finally, a set of hand-crafted rules is used to match entities. For example, "Inc." frequently occurs in organization names, so the appearance of "Inc." indicates that the token should be labeled as ORG. The entire process is summarized in Figure A1.



```
SELECT ?item   WHERE   {
wd: Q16977 wdt:P31/wdt:P279* ?item.
FILTER ( ?item in (Q1733492) )  }
```

```
SELECT ?item   WHERE   {
wd: Q16977 wdt:P31/wdt:P279* ?item.
FILTER ( ?item in (Q215627) )  }
```

Figure A1. Illustration of the KB-Match method [22].

Appendix B. Greedy Sampling Method

Sampling only K sentences for each entity type may lead to a bias towards frequent entity types. Therefore, a greedy sampling strategy is proposed in [5], and is reproduced in Algorithm A1. This strategy ensures that a roughly equal number of entities are sampled for each entity type.

Algorithm A1 Greedy sampling.

```
Require: The Original dataset $\{(X_m, Y_m)\}_{m=1}^{M}$; The number of $K$; entity type set $C'$ .
  sort type in $C'$ based on their frequency in The Original dataset
  $D_{train} \leftarrow \emptyset$ //Init the Train Dataset
  $Count_i \leftarrow 0$ // Init counts if entity type in Train Dataset
  while $i < \left|C'\right|$ do
    while $Count_i < K$ do
      sample $(x, y) \in \{(X_m, Y_m)\}_{m=1}^{M}$
      $D_{train} \leftarrow D_{train} \cup \{(x, y)\}$
      Update $\{Count_j\} \quad \forall C_j \in y$
    end while
  end while
Output: few-shot dataset $D_{train}$
```

References

1. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Chapter of the Association for Computational Linguistics: Human Language Technologies, (Long and Short Papers), Proceedings of the 2019 Conference of the North American, Cambridge, MA, USA, 8–11 November 2019*; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; Volume 1, pp. 4171–4186.
2. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
3. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 7871–7880.
4. Fritzler, A.; Logacheva, V.; Kretov, M. Few-shot classification in named entity recognition task. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, 8–12 April 2019; pp. 993–1000.
5. Yang, Y.; Katiyar, A. Simple and Effective Few-Shot Named Entity Recognition with Structured Nearest Neighbor Learning. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; pp. 6365–6375.
6. Huang, J.; Li, C.; Subudhi, K.; Jose, D.; Balakrishnan, S.; Chen, W.; Peng, B.; Gao, J.; Han, J. Few-shot named entity recognition: An empirical baseline study. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Online, 10 May 2021; pp. 10408–10423.
7. Hou, Y.; Che, W.; Lai, Y.; Zhou, Z.; Liu, Y.; Liu, H.; Liu, T. Few-shot Slot Tagging with Collapsed Dependency Transfer and Label-enhanced Task-adaptive Projection Network. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 1381–1393.
8. Ma, J.; Ballesteros, M.; Doss, S.; Anubhai, R.; Mallya, S.; Al-Onaizan, Y.; Roth, D. Label Semantics for Few Shot Named Entity Recognition. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, 22–27 May 2022; pp. 1956–1971.
9. Ma, T.; Jiang, H.; Wu, Q.; Zhao, T.; Lin, C.Y. Decomposed Meta-Learning for Few-Shot Named Entity Recognition. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, 22–27 May 2022; pp. 1584–1596.
10. Wang, Y.; Chu, H.; Zhang, C.; Gao, J. Learning from Language Description: Low-shot Named Entity Recognition via Decomposed Framework. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2021, Punta Cana, Dominican Republic, 17 April 2021; pp. 1618–1630.
11. Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.* **2023**, *55*, 1–35. [CrossRef]
12. OpenAI. ChatGPT. Available online: https://openai.com/blog/chatgpt/ (accessed on 8 March 2023).
13. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
14. Cui, L.; Wu, Y.; Liu, J.; Yang, S.; Zhang, Y. Template-Based Named Entity Recognition Using BART. In Proceedings of the Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, Online, 1–6 August 2021; pp. 1835–1845.
15. Chen, X.; Zhang, N.; Li, L.; Xie, X.; Deng, S.; Tan, C.; Huang, F.; Si, L.; Chen, H. Lightner: A lightweight generative framework with prompt-guided attention for low-resource ner. *arXiv* **2021**, arXiv:2109.00720.
16. Ma, R.; Zhou, X.; Gui, T.; Tan, Y.; Li, L.; Zhang, Q.; Huang, X. Template-free Prompt Tuning for Few-shot NER. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Seattle, WA, USA, 10–15 July 2022; pp. 5721–5732.

17. Ding, N.; Chen, Y.; Han, X.; Xu, G.; Wang, X.; Xie, P.; Zheng, H.; Liu, Z.; Li, J.; Kim, H.G. Prompt-learning for Fine-grained Entity Typing. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 6888–6901.
18. Das, S.S.S.; Katiyar, A.; Passonneau, R.; Zhang, R. CONTaiNER: Few-Shot Named Entity Recognition via Contrastive Learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Association for Computational Linguistics: Dublin, Ireland, 2022; pp. 6338–6353.
19. Gao, T.; Fisch, A.; Chen, D. Making Pre-trained Language Models Better Few-shot Learners. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online, 22–27 May 2021; pp. 3816–3830.
20. Shang, J.; Liu, L.; Ren, X.; Gu, X.; Ren, T.; Han, J. Learning named entity tagger using domain-specific dictionary. *arXiv* **2018**, arXiv:1809.03599.
21. Peng, M.; Xing, X.; Zhang, Q.; Fu, J.; Huang, X. Distantly supervised named entity recognition using positive-unlabeled learning. *arXiv* **2019**, arXiv:1906.01378.
22. Liang, C.; Yu, Y.; Jiang, H.; Er, S.; Wang, R.; Zhao, T.; Zhang, C. BOND: Bert-Assisted Open-Domain Named Entity Recognition with Distant Supervision. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Online, 6–10 July 2020.
23. Meng, Y.; Zhang, Y.; Huang, J.; Wang, X.; Zhang, Y.; Ji, H.; Han, J. Distantly-supervised named entity recognition with noise-robust learning and language model augmented self-training. *arXiv* **2021**, arXiv:2109.05003.
24. Ying, H.; Luo, S.; Dang, T.; Yu, S. Label Refinement via Contrastive Learning for Distantly-Supervised Named Entity Recognition. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, USA, 10–15 June 2022; pp. 2656–2666.
25. Jiang, Z.; Xu, F.F.; Araki, J.; Neubig, G. How Can We Know What Language Models Know? *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 423–438. [CrossRef]
26. Shin, T.; Razeghi, Y.; Logan IV, R.L.; Wallace, E.; Singh, S. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; pp. 4222–4235.
27. Liu, X.; Zheng, Y.; Du, Z.; Ding, M.; Qian, Y.; Yang, Z.; Tang, J. GPT Understands, Too. *arXiv* **2021**, *arXiv:2103.10385*.
28. Li, X.L.; Liang, P. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online, 22–27 May 2021; pp. 4582–4597.
29. Lester, B.; Al-Rfou, R.; Constant, N. The Power of Scale for Parameter-Efficient Prompt Tuning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Online, 7–11 November 2021; pp. 3045–3059.
30. Qin, G.; Eisner, J. Learning How to Ask: Querying LMs with Mixtures of Soft Prompts. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; pp. 5203–5212.
31. Hu, S.; Ding, N.; Wang, H.; Liu, Z.; Wang, J.; Li, J.; Wu, W.; Sun, M. Knowledgeable Prompt-tuning: Incorporating Knowledge into Prompt Verbalizer for Text Classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Association for Computational Linguistics: Dublin, Ireland, 2022; pp. 2225–2240.
32. Min, S.; Lewis, M.; Hajishirzi, H.; Zettlemoyer, L. Noisy Channel Language Model Prompting for Few-Shot Text Classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Association for Computational Linguistics: Dublin, Ireland, 2022; pp. 5316–5330.
33. Li, C.; Gao, F.; Bu, J.; Xu, L.; Chen, X.; Gu, Y.; Shao, Z.; Zheng, Q.; Zhang, N.; Wang, Y.; et al. SentiPrompt: Sentiment Knowledge Enhanced Prompt-Tuning for Aspect-Based Sentiment Analysis. *arXiv* **2021**, arXiv:2109.08306.
34. Han, X.; Zhao, W.; Ding, N.; Liu, Z.; Sun, M. PTR: Prompt Tuning with Rules for Text Classification. *AI Open* **2021**, *3*, 182–192. [CrossRef]
35. Sainz, O.; Lopez de Lacalle, O.; Labaka, G.; Barrena, A.; Agirre, E. Label Verbalization and Entailment for Effective Zero and Few-Shot Relation Extraction. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Online, 7–11 November 2021; pp. 1199–1212.
36. Tan, Z.; Zhang, X.; Wang, S.; Liu, Y. MSP: Multi-Stage Prompting for Making Pre-trained Language Models Better Translators. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Association for Computational Linguistics: Dublin, Ireland, 2022; pp. 6131–6142.
37. Wang, S.; Tu, Z.; Tan, Z.; Wang, W.; Sun, M.; Liu, Y. Language Models are Good Translators. *arXiv* **2021**, arXiv:2106.13627.
38. Schick, T.; Schütze, H. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Online, 19–23 April 2021; pp. 255–269.
39. Schick, T.; Schütze, H. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 4 November 2021; pp. 2339–2352.

40. Gu, Y.; Han, X.; Liu, Z.; Huang, M. PPT: Pre-trained Prompt Tuning for Few-shot Learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Association for Computational Linguistics: Dublin, Ireland, 2022; pp. 8410–8423.
41. Zheng, Y.; Zhou, J.; Qian, Y.; Ding, M.; Liao, C.; Jian, L.; Salakhutdinov, R.; Tang, J.; Ruder, S.; Yang, Z. FewNLU: Benchmarking State-of-the-Art Methods for Few-Shot Natural Language Understanding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Association for Computational Linguistics: Dublin, Ireland, 2022; pp. 501–516.
42. Wu, Q.; Lin, Z.; Wang, G.; Chen, H.; Karlsson, B.F.; Huang, B.; Lin, C.Y. Enhanced meta-learning for cross-lingual named entity recognition with minimal resources. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 7–12 February 2020; Volume 34, pp. 9274–9281.
43. Jiang, H.; Zhang, D.; Cao, T.; Yin, B.; Zhao, T. Named Entity Recognition with Small Strongly Labeled and Large Weakly Labeled Data. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online, 1–6 August 2021; pp. 1775–1789.
44. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online, 16–20 November 2020; pp. 38–45.
45. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. In Proceedings of the International Conference on Learning Representations, Boston, MA, USA, 20–23 August 2017.
46. Tjong Kim Sang, E.F.; De Meulder, F. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL, Edmonton, AB, Canada, 31 May 2003; pp. 142–147.
47. Weischedel, R.; Palmer, M.; Marcus, M.; Hovy, E.; Pradhan, S.; Ramshaw, L.; Xue, N.; Taylor, A.; Kaufman, J.; Franchini, M.; et al. OntoNotes Release 5.0 LDC2013T19. Web Download; Linguistic Data Consortium: Philadelphia, PA, USA, 2013. [CrossRef]