Inverse is Better! Fast and Accurate Prompt for Slot Tagging

Anonymous ACL submission

Abstract

Prompting methods recently achieve impressive success in few-shot learning. These methods embed input samples with prompt sentence pieces and decode label-related tokens to map samples to the label. However, such a paradigm is very inefficient for the task of slot tagging. Because the slot tagging samples are multiple consecutive words in a sentence, the prompting methods have to enumerate all n-grams token span to find all the possible slots, which greatly slows down the prediction. To tackle this, we introduce an inverse paradigm for prompting. Different from the classic prompts map tokens to labels, we reversely predict slot values given slot types. Such inverse prompting only requires a one-turn prediction for each slot type and greatly speeds up the prediction. Besides, we propose a novel Iterative Prediction Strategy, from which the model learns to refine predictions by considering the relations between different slot types. We find, somewhat surprisingly, the proposed method not only predicts faster, but also significantly improves the effect (improve over 6.1 F1-scores on 10-shot setting) and achieves new state-of-the-art performance.

1 Introduction

Few-shot learning (FSL) aims at learning a model from only a few examples and is regarded as one of the key steps toward more human-like artificial intelligence (Wang et al., 2020). Recently, prompt-based methods achieve impressive results and show promising prospects for few-shot learning of Natural Language Processing (NLP) (Liu et al., 2021a; Zhao et al., 2021).

Prompt-based methods reformulate a target task into the language modeling problem, which takes advantages of the powerful Pretrained Language Models (LM) (Devlin et al., 2019; Liu et al., 2019; Lewis et al., 2020; Brown et al., 2020). For example, when classifying the sentiment of the movie review “no reason to watch”, prompting methods insert a piece of text "It was", i.e. prompts, to the input examples, getting “No reason to watch. It was __”. It is natural to expect a higher probability from the LM to fill the template with "terrible" than “great”, and the original task is then converted to a language modeling task. Such conversion reduces the gap between pretraining and target tasks, which allows depending less on target task data and helps to achieve better performance in low data scenarios (Gao et al., 2021).

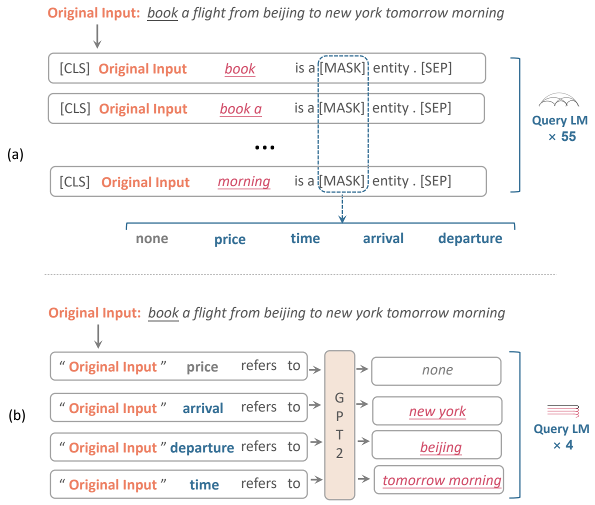

Figure 1: An example of normal (a) and inverse (b) prompting methods for slot tagging. For normal prompts, identifying all slots in the query sentence requires enumeration of all spans, while inverse prompt only needs 1-time prediction for each label.

However, while achieving great success in sentence-level tasks, prompting-based methods show incompatibility for sequence labeling task, such as slot tagging. Firstly, the aforementioned prompting paradigm is quite inefficient for slot tagging task. Different from the sentence-level tasks that classify samples of whole sentences, slot tagging samples are multiple consecutive words in a sentence. Therefore, as shown in Figure 1, to find all the possible slots, prompt-based methods have to enumerate all n-gram word spans, and then query LM for each of them, which greatly slows down the prediction (Cui et al., 2021). Further, as a structure prediction problem, slot tagging benefits from taking the dependencies between labels into account (Ma and Hovy, 2016; Hou et al., 2020) For example in Figure 1, where `to.Loc` entity often appear after `from.Loc` entity. Such label dependency is hard to be captured by current prompting methods, since they predict labels one-by-one independently.

To tackle the above issues, we introduce an inverse paradigm for prompting. Different from the classic prompts map tokens to labels, we reversely predict slot values given slot types. For the example in Figure 1, we embed the input with an inverse prompt as "*book a flight from Beijing to New York tomorrow morning*. arrival *refers to* __", and then LM is able to decode multi-word span "New York" at a time. Compared to the classic prompts that require predictions for every n-gram word span (55-times in Figure 1), we only need to perform decoding for V-times, where V is the number of label types (4-times in Figure 1), and therefore greatly speed up the prediction. Surprisingly, experiments show the proposed method not only predicts faster, but also significantly improve the performance, indicating that prompting LM reversely is a better fit for the slot tagging task. Besides, to further improve the prediction accuracy, we propose a novel Iterative Prediction Strategy, from which the model learns to refine predictions by considering the relations between different slot types.

To summarize the contribution of this work:

(1) We introduce the idea of inverse prediction to prompting-methods for slot tagging task, which greatly speeds up the prediction process.

(2) We propose an Iterative Prediction Strategy for learning and prediction for slot tagging prompt, which allows the prompting model to consider dependency between different slot types and refine prediction.

(3) We extensively evaluate the proposed method in various few-shot settings, where the proposed method brings significant improvements not only for the speed, but also for the accuracy.

All code and data will be publicly available.

2 Background

In this part, we first present a formal definition of the few shot slot tagging task in Section 2.1, followed by an introduction of the conventional sequence labeling approaches in Section 2.2 and Sequence Labeling with Prompts in Section 2.3.

2.1 Few Shot Slot Tagging

We define sentence $\boldsymbol{x} = (x_1, x_2, ...x_n)$ as a sequence of words and $\boldsymbol{y} = (y_1, y_2, ..., y_n)$ as the label sequence matching the sentence $\boldsymbol{x}$, a domain $D = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N_D}$ is a set of $(\boldsymbol{x}, \boldsymbol{y})$, and the label set $L_D = \{l_i\}_{i=1}^{N_{L_D}}$ is unique to each domain.

In few shot scenarios, there are a set of low-resource domains $\{D_L^{(1)}, D_L^{(2)}, ...\}$ called target domains. Each target domain $D_L^{(j)}$ only contains a few labeled instances called support set $S = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N_S}$, which usually includes K examples (K-shot) for each of N labels (N-way). On each target domain, given support set examples as references, few shot slot tagging models are required to make predictions for query set samples. Optionally, some few-shot settings also include a set of optional rich-data domains $\{D_H^{(1)}, D_H^{(2)}, ...\}$ called source domains, which are used for pretraining of few-shot models.

2.2 Conventional Sequence Labeling Approaches

Conventional approaches regard slot tagging as a sequence labeling problem where each word in a sentence is assigned with a BIO-based label. For the example in the Figure 2, `B-time` is tagged to the first word in a time slot, `I-time` is tagged to a non-begin word within a time slot, and `O` label refers to non-slot tokens. Few-shot slot tagging is then defined as: given a K-shot support set S and an input query sequence $\boldsymbol{x} = (x_1, x_2, ..., x_n)$, find $\boldsymbol{x}$'s best label sequence $\boldsymbol{y}^*$. As shown in Figure 2(a), this method can be formulated as:

$$\begin{gathered}
\boldsymbol{h_{1:n}} = \text{Encoder}(\boldsymbol{x_{1:n}}), \\
p(\boldsymbol{y_c}|\boldsymbol{x}, S) = \text{Softmax}(\text{Decoder}(\boldsymbol{h_c})), \\
(c \in [1, 2, ..., n]), \\
\boldsymbol{y}^* = (y_1, y_2, ..., y_n) = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}, S),
\end{gathered}$$

where Encoder is usually a pretrained language model such as BERT (Devlin et al., 2019), $h_{1:n}$ is the hidden state of the encoder with a dimension d_h, Decoder can be a linear layer, a CRF layer or any other parametric or non-parametric classifier.

2.3 Sequence Labeling with Prompts

Prompt-based methods have been proven effective in many NLU tasks especially in few-shot settings,

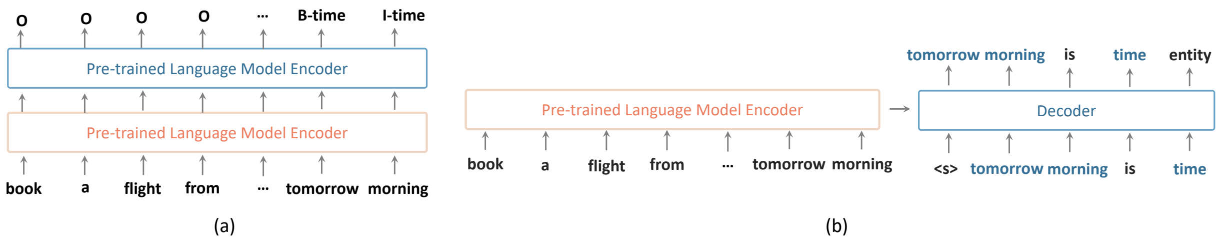

Figure 2: Illustration of conventional sequence labeling method (a) and classic prompting methods (b)

but things become complicated when it comes to slot tagging tasks. In Cui et al. (2021), a slot $s_j^i = \{x_i, ..., x_j\}$ is a span starts from x_i and ends with x_j, and they construct a template *"[x_i] [s_i^j] is a [z] entity."* to predict *[z]* (e.g., person) corresponding to an entity label (e.g., PERSON) after finetuned on this kind of template support set. In their method, to construct templates, we need to traverse all the n-gram spans $s_j^i, i, j \in [1, n]$ in a sentence with each label in the label set which is quite expensive in time and compute resources.

3 Method

In this section, we propose a new paradigm for few-shot slot tagging using an inverse prompt to convert slot tagging into a generation task. We first introduce how to create our reverse prompts in Section 3.1, then show the inference details in Section 3.2 and the Iterative Prediction Strategy in Section 3.3, respectively.

3.1 Prompt Creation

We create the inverse prompt P and turn slot tagging into a generation task by filling a template combined with input text and slot labels. Our prompt P consists of two parts, i,e., the label mapping and the inverse prompt template.

The label mapping is a one-to-one mapping function to convert the label set $L = \{l_1, ..., l_{|L|}\}$ (e.g., l_k = `to.Loc`) to a natural word set $\hat{L} = \{\hat{l_1}, ..., \hat{l_{|L|}}\}$ (e.g. $\hat{l_k}$ = `departure`). And the inverse prompt templates are constructed by querying each label in the label set for a given original sentence. Specifically, given an input original sentence s and a set of labels $\hat{L} = \{\hat{l_i}\}$, for each label $\hat{l_i} \in \hat{L}$, our prompted inputs are defined as:

$$\text{"}s\text{"}\ \hat{l_i}\ \textit{refers to}\ \_\_,$$

and the model requires to generate slot values naturally. By guiding the language model to continue generating slot values naturally, we leverage knowledge from pretrained language models to our slot tagging tasks.

3.2 Reverse Inference with Prompts

In this section, we will introduce how the generative slot tagging is conducted in the inference procedure with proposed inverse prompts.

The inference procedure can be concluded as the following steps: (1) We use the label mapping to map all labels $\{l_1, ..., l_{|L|}\}$ in the label set to $\{\hat{l_1}, ..., \hat{l_{|L|}}\}$. (2) For each mapped label $\hat{\boldsymbol{l}}_j$, we sample one input sentence $\boldsymbol{s}_i$, then fill them in the prepared template to get prompted input $\boldsymbol{x}_{ij}$. (3) We use the fine-tuned pre-trained language model to conduct a controlled generation procedure in which generation word-list is constraint in the original sentence along with structure control tokens $\boldsymbol{t} \in \hat{\boldsymbol{s}}_i = \boldsymbol{s}_i \cup \{$`<NONE>`, `<SEP>`, `<END>`$\}$. Specially, for the control tokens, we use *"none"* as `<NONE>` token if there's no corresponding slot value in $\boldsymbol{s}$; we use *";"* as `<SEP>` token to divide more than one corresponding slot values and we use "." as `<END>` token to indicate the end of the generation. For each prompted input $\boldsymbol{x_{ij}}$, the next token t_k is determined by:

$$\boldsymbol{t}_k = \underset{\boldsymbol{t}_k \in \hat{\boldsymbol{s}}_i}{\arg\max}\, log(\boldsymbol{p}(\boldsymbol{t}_k | \boldsymbol{x}_{ij}; \boldsymbol{t}_{1:k-1}))$$

As shown in Figure 3, given a sentence *'book a flight from beijing to new york tomorrow morning'* and a label set L = {`from.Loc`, `to.Loc`, `Time`, `Price`}. (1) We map the label to a natural language label set $\hat{L}$ = { `departure`, `arrival`, `time`, `price`}. (2) For each $\hat{l} \in \hat{L}$, we fill them into the template to get prompted inputs. (3) We feed the prompted inputs into our model to generate corresponding slot values following the text-generation procedure until reaching the max length or having a full stop generated.

3.3 Iterative Prediction Strategy

The Iterative Prediction Strategy completes the whole prediction process by revising the slot values

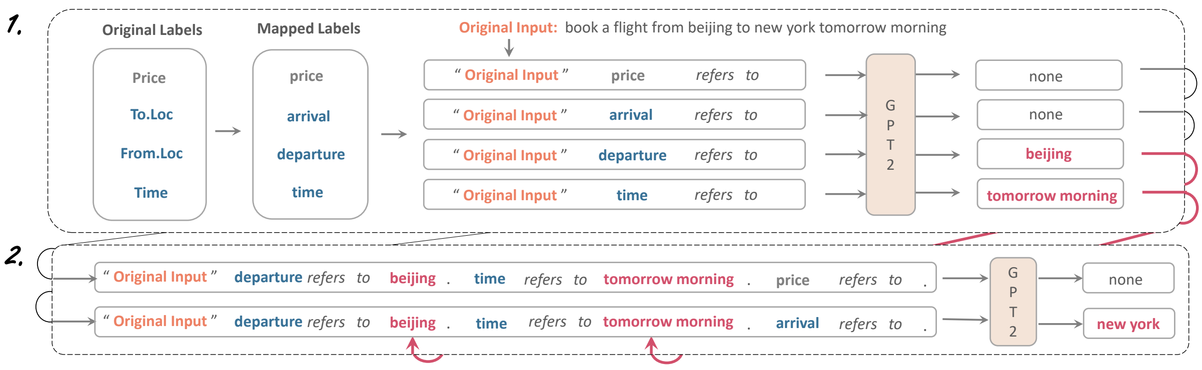

Figure 3: Overview of propose method with Inverse Prediction and Iterative Prediction Strategy.

that were "*none*" in the first iteration. We assume that different labels are interactive, so the predicted slot values could be used as a hint to help predict those "*none*" ones. For example, the model tends to successfully generate the slot value of "arrival" given the results of "departure" and "time" in the first iteration (Figure 3). Motivated by this, we construct another template for the Iterative Prediction Strategy, which concatenates those predicted prompts and places them before the unpredicted prompted inputs. Below we introduce the strategy for the inference and training stages in detail.

At the **inference** time, as shown in Figure 3, we take the predicted slot labels in the first round into inputs for models to process. We denote the original input as s, and the i-th recognized labels in the first iteration as $l_i^r \in L_R$, the j-th unpredicted labels (whose slot values are "*none*") as $l_j^u \in L_U (L_U = \hat{L} \setminus L_R)$. So for unrecognized slot label l_j^u the prompted inputs are constructed as:

"s" l_1^r refers to <slot_value$_1$> . ... . l_n^r refers to <slot_value$_n$> . l_j^u refers to__.

The model revises the unrecognized slots given the above prompted inputs during the second iteration.

During the **training** time, we simulate the cases where the slots are not recognized so as to enable the model to revise the *none* slot values. We do this by manually constructing *none* slot value examples. Specifically, for each original sentence s, we randomly select some occurred labels l^s (e.g., "arrival" in Fig. 3) and combine them with the non-occurred labels (e.g., "price" in Fig. 3) to construct the unrecognized set L_U. The rest of the occurred labels (e.g., "departure" and "time" in Fig. 3) form the recognized set L_R.

Given the i-th recognized slot label $l_i^r \in L_R$ and the j-th unrecognized slot label $l_j^u \in L_U$, the prompted inputs are constructed as follows:

"s" l_1^r refers to <slot_value$_1$> . ... . l_n^r refers to <slot_value$_n$> . l_j^u refers to__

The model outputs *"none"* if l_j^u is from the non-occurred labels. It outputs the corresponding slot values if l_j^u is the selected label l^s (If multiple slot values are generated, we separate them with *";"*).

4 Experiment

We evaluate the performance of the proposed method on two types of few-shot learning benchmarks: (1) Setting with Only In-domain data, where all training data are only a few labeled support data. (2) Setting with Meta Source Tasks, where some additional data-rich source domains are available for pretraining.

Evaluation To directly compare with conventional sequence labeling methods, we need to label tokens reversely. After generation, we first separate outputs into slot values. For each slot value, we label tokens in the source sentence with three principles: (1) Slot value is complete: only if the whole slot value matches a span in the source sentence, we label it with the corresponding label. (2) Choose the first overlap predicted slot span: if any token in the source sentence has been labeled, we do not relabel this token even when it matches another slot value. (3) Use BIO labels: add 'B-' to the beginning token of the slot span, add 'I-' to the non-begin token of the slot span and label non-slot tokens with 'O'. After labeling tokens reversely, we evaluate F1 scores within each few-shot episode.[1]

[1]For each episode, we calculate the F1 score on query samples with `conlleval` script: https://www.clips.uantwerpen.be/conll2000/chunking/conlleval.txt

4.1 Setting with Only In-domain data

Datasets To evaluate our proposed method in only few-shot in-domain data without source domain knowledge transfer, we conduct experiments on three few-shot datasets: MIT-Restaurant Review (Liu et al., 2013), MIT-Movie Review (Liu et al., 2013) and MIT-Movie-Hard. [2] Each dataset has 10 episodes, and each episode consists of a different k-shot support set and the same query set.

Implements We conduct experiments with $K \in \{10, 20, 50, 100, 200, 500\}$ shot few-shot settings to fully evaluate the performance of our method in all three datasets. Our proposed method employs GPT2-small (Radford et al., 2019) pre-trained model as the base model for fine-tuning, and no new parameters are introduced. Besides, we set the learning rate=$6.25e-5$ and batch size=2 for few-shot training. For all our experiments, we fine-tune the model only on few-shot support set for 2~4 epochs with the AdamW optimizer and linear decaying scheduler.

Baselines In our experiments, we provide competitive conventional sequence labeling method, forward template-based method and some methods pretrained on data-rich source domains.

- **Sequence Labeling BERT** (Devlin et al., 2019) can be seen as a BERT-based sequence labeling baseline which fine-tunes the BERT model with a token-level linear classifier head.
- **Template-based BART** (Cui et al., 2021) uses BART to encode the source sentence and decodes the template constructed by querying each possible span in a sentence with each class separately.
- **NNShot and StructShot** (Yang and Katiyar, 2020) are two metric-based few-shot learning approaches for slot tagging and NER. NNShot is an instance-level nearest neighbor classifier for few-shot prediction, and StructShot promotes NNShot with a Viterbi algorithm during decoding.
- **EntLM** (Ma et al., 2021b) is a prompt-based method using one pass language model replacing label words with pre-selected slot values.

Results Table 1 shows the results of the proposed method only finetuned on few-shot in-domain data and baselines under few-shot setting. Among these methods, we can observe that:

(1) Our proposed method performs consistently better than all the baseline methods on all three datasets. It outperforms the strongest baseline Template-based BART which uses BART-large by average F1 scores on three datasets of 11.96 in 10-shot setting even with a 40% smaller pretrained language model GPT2-small.

(2) Our proposed method is even comparable or outperforms those baselines with data-rich domain pretrained.

(3) Our proposed method performs much better than baselines in fewer labeled samples settings, especially in 10 and 20 shot settings, which indicates our method can leverage information from limited labeled data more efficiently.

(4) Our method significantly outperformed Sequence Labeling BERT whose performance is quite poor on 10 and 20 shot settings, which indicates that the number of labeled data under the few-shot setting is scarce for conventional sequence labeling task, and proves that the prompt-based method is effective in few-shot slot tagging tasks.

(5) The proposed Iterative Prediction Strategy improves our method by average F1 score on three datasets of 2.23 and 1.44 in 10 and 20 shot setting respectively and even sees improvements in 200 and 500 shot settings, which proves the effectiveness of the Iterative Prediction Strategy in very few labeled data settings and it may still work in middle size labeled data scenarios.

4.2 Setting with Meta Source Task

Datasets To evaluate the transferability from data-rich domains to unseen few-shot domains of our proposed model, we conduct experiments on SNIPS (Coucke et al., 2018) dataset. Following the data split provided by Hou et al. (2020), we construct 5-shot SNIPS datasets from the original SNIPS datasets. The few-shot SNIPS dataset consists of 7 domains with different label sets: GetWeather (We), Music (Mu), PlayList (Pl), RateBook (Bo), SearchScreenEvent (Se), BookRestaurant (Re) and SearchCreativeWork (Cr). Each domain contains 100 episodes, and each episode consists of a support set with a batch of labeled samples and query samples to evaluate.

Implements Following Henderson and Vulic (2021), we conduct our cross-domain experiments with 5-shot few-shot settings to evaluate the ability of our model to transfer from rich-data domains to unseen few-shot domains. For our proposed

[2] MIT-Movie Review has two datasets, the simple and the complex. We regard the simple one as MIT-Movie and combine both as MIT-Movie-Hard.

Model	MIT-Restaurant					
	10	**20**	**50**	**100**	**200**	**500**
Wiseman and Stratos (2019) + PT	4.1	3.6	4.0	4.6	5.5	8.1
Ziyadi et al. (2020) + PT	27.6	29.5	31.2	33.7	34.5	34.6
Huang et al. (2020) + PT	46.1	48.2	49.6	50.0	50.1	
Sequence Labeling BART + PT	8.8	11.1	42.7	45.3	47.8	58.2
Sequence Labeling BERT + PT	27.2	40.9	56.3	57.4	58.6	75.3
Template-based BART + PT	**53.1**	60.3	64.1	67.3	72.2	75.7
Sequence Labeling BERT	21.8	39.4	52.7	53.5	57.4	61.3
Template-based BART	46.0	57.1	58.7	60.1	62.8	65.0
Ours	49.35	60.48	65.34	70.41	73.69	76.13
Ours + Iterative	**52.10**	**61.49**	**66.83**	**70.98**	**73.97**	**76.37**
Model	**MIT-Movie-Hard**					
	10	**20**	**50**	**100**	**200**	**500**
Wiseman and Stratos (2019) + PT	3.1	4.5	4.1	5.3	5.4	8.6
Ziyadi et al. (2020) + PT	40.1	39.5	40.2	40.0	40.0	39.5
Huang et al. (2020) + PT	36.4	36.8	38.0	38.2	35.4	38.3
Sequence Labeling BART + PT	13.6	30.4	47.8	49.1	55.8	66.9
Sequence Labeling BERT + PT	28.3	45.2	50.0	52.4	60.7	76.8
Template-based BART + PT	42.4	54.2	59.6	65.3	69.6	**80.3**
Sequence Labeling BERT	25.2	42.2	49.64	50.7	59.3	74.4
Template-based BART	37.3	48.5	52.2	56.3	62.0	74.9
Ours	52.07	59.11	65.63	69.35	72.36	**75.03**
Ours + Iterative	**53.31**	**60.19**	**66.13**	**69.63**	**72.45**	74.83
Model	**MIT-Movie**					
	10	**20**	**50**	**100**	**200**	**500**
Sequence Labeling BERT	50.60	59.34	71.33	-	-	-
NNShot	50.47	58.94	71.17	-	-	-
StructShot	53.19	61.42	72.07	-	-	-
Template-based BART	49.30	59.09	65.13	-	-	-
EntLM	49.30	59.09	65.13	-	-	-
Ours	57.04	67.86	76.81	80.28	82.43	**84.55**
Ours + Iterative	**59.74**	**70.09**	**77.60**	**80.63**	**82.64**	84.51

Table 1: F1 scores of few-shot slot tagging task on three different datasets.10 indicates 10 instances for each entity types. **+PT** denotes using model are pretrained on additional datasets. **+Iterative** denotes enhance model with Iterative Prediction Strategy.

method, same as in-domain settings, we use GPT2-small pre-trained model as the base model for pre-training in source domain and fine-tuning in target few-shot domain, and no new parameters are introduced. We set learning rate=$6.25e-5$ and batch size=16 for pretraining and batch size=2 for 5-shot finetuning. During finetuning, we use the same AdamW optimizer and linear decaying scheduler. The hyper-parameters are decided according to performance on the dev set.

Baselines We provided competitive strong baselines, including traditional methods, finetune-based methods and advanced few-shot learning methods.

• **Bi-LSTM** (Schuster and Paliwal, 1997) uses GLoVe (Pennington et al., 2014) embedding for slot tagging. Train on the support sets and test on the query examples.

• **SimBERT** is a metric-based method using original BERT to label tokens with the most similar token's label in cosine similarity.

• **Matching Network (MN)** (Vinyals et al., 2016) A few-shot sequence labeling model employing the matching network with BERT embedding for token-level classification.

• **TransferBERT** is a domain transfer conventional NER model using BERT, pretrained on source domains and fine-tuned on target domain support set and performs on the test set

• **WPZ** (Fritzler et al., 2019) is a metric-based few-shot slot tagging method using the prototypical network (Snell et al., 2017). It pre-trains a prototypical network on source domains, and utilizes the network to do word-level classification on target domains without training.

• **TapNet+CDT**, **L-TapNet+CDT**, **L-WPZ+CDT** (Hou et al., 2020) are advanced metric-based few-shot learning methods, using a CRF framework based on source domain pretrained BERT to predict label in target domain without further training.

• **ConVEx** (Henderson and Vulic, 2021) is a fine-tuning based method that models slot tagging as a

Model	5-shot Slot Tagging							
	We	Mu	Pl	Bo	Se	Re	Cr	Ave.
Bi-LSTM	25.44	39.69	45.36	73.58	55.03	40.30	40.49	45.70
SimBERT	53.46	54.13	42.81	75.54	57.10	55.30	32.38	52.96
TransferBERT	56.01	43.85	50.65	14.19	23.89	36.99	14.29	34.27
MN	38.80	37.98	51.97	70.61	37.24	34.29	72.34	49.03
WPZ+BERT	69.06	57.97	44.44	71.97	74.62	51.01	69.22	62.61
TapNet+CDT	67.83	68.72	73.74	86.94	72.12	69.19	66.54	72.15
L-WPZ+CDT	**78.23**	62.36	59.74	76.19	83.66	69.69	71.51	71.62
L-TapNet+CDT	69.58	64.09	74.93	85.37	**83.76**	69.89	73.80	74.49
ConVEx*	71.5	77.6	**79.0**	84.5	84.0	**73.8**	67.4	76.8
Ours	70.44	71.63	78.67	**87.37**	81.38	71.77	74.42	76.53
Ours + Iterative	70.63	**71.97**	**78.73**	87.34	**81.95**	**72.07**	**74.44**	**76.73**

Table 2: F1 score results on 5-shot Snips. Our methods achieve the best performance. * denotes using additional Reddit data for pretraining.

cloze task first pre-trained on Reddit data to learn general span extraction ability, then fine-tuned on few-shot slot tagging data. Note that the Reddit data is not used by our method and other baselines during the experiment.

Results Table 2 shows the results of the cross-domain few-shot setting. Among these methods in the table, we can observe that:

(1) Our proposed method outperforms all the baselines except ConVEx which uses extra Reddit data in cross-domain 5-shot setting.

(2) We outperform TransferBERT by 42.36 F1 scores which strongly proved that prompt-based method can transfer more knowledge from source domain and more data-efficient than conventional methods. Noting that we can directly compare with TransferBERT for both our methods first pre-trained on source domains and then finetuned on each few-shot domain respectively without any few-shot learning tricks.

(3) Our method outperforms some metric-based few-shot learning baselines, for example, 2.24 F1 scores higher than L-TapNet+CDT, which demonstrate the effectiveness of prompt method in the slot tagging task.

(4) Our Iterative Prediction Strategy improved Our method by about 0.5 F1 scores, demonstrating its effectiveness under cross-domain scenarios.

4.3 Analysis

Effects of Iterative Prediction Learning As shown in Table 1, the proposed Iterative Prediction Learning brings consistent improvement ,especially in low-resource settings. It works by revising predictions with a second-round query to recognize those missing slots, which can bring an increase in recall score. To confirm that, we make our analysis about precision score (P), recall score (R) and F1 score (F), as shown in Table 3.

	Model	Restaurant			Movie		
		P	R	F	P	R	F
10	Ours	67.7	**42.4**	**52.1**	84.0	**46.4**	**59.7**
	w/o Iter	69.4	38.3	49.4	85.9	42.7	57.0
	w/o Joint	68.8	38.9	49.7	85.6	43.0	57.2
20	Ours	70.1	**54.7**	**61.5**	83.5	**60.4**	**70.1**
	w/o Iter	71.6	52.3	60.5	86.3	55.9	67.9
	w/o Joint	70.92	53.45	61.0	85.6	56.9	68.3
50	Ours	73.6	**61.2**	**66.8**	83.6	**72.4**	**77.6**
	w/o Iter	75.4	57.6	65.3	85.9	69.5	76.8
	w/o Joint	74.3	59.2	65.7	84.7	70.8	77.1
100	Ours	76.1	**66.5**	**71.0**	84.4	**77.2**	**80.6**
	w/o Iter	78.0	64.2	70.4	86.3	75.0	80.3
	w/o Joint	76.7	66.0	71.0	85.0	76.5	80.5
200	Ours	77.8	**70.5**	**74.0**	85.4	**80.0**	**82.6**
	w/o Iter	79.5	68.7	73.7	87.1	78.2	82.4
	w/o Joint	78.0	70.1	73.8	85.1	79.9	82.4
500	Ours	79.4	**73.5**	**76.4**	86.3	**82.8**	84.5
	w/o Iter	81.0	71.8	76.1	87.9	81.4	**84.6**
	w/o Joint	79.6	73.4	76.4	86.6	82.1	84.3

Table 3: Ablation analysis Iterative Prediction Strategy **w/o Iter** denotes removing iterative strategy and **w/o joint** denotes using two separate models for the two iterative steps.

When Iterative Revise Learning is added, we can get a rise in recall score about 4 percent in 10-shot, 2~4 percent in 20 shot and more than 1 percent in other shot settings in exchange for a slight precision drop, resulting in a rise in overall F1 score by about 2 percent in 10 and 20 shots.

We further explore whether a sequential jointly trained model from first-round training or a from-scratch training model in Iterative Prediction Strategy training time performs better by conducting experiment training from scratch. As shown in Table 3, without jointly training, the revised performance drops, but still brings improvements, which further proves the effectiveness of proposed Iterative Prediction Strategy.

Model	Movie	Restaurant
Baseline (Normal Prompt)	408.0	236.0
Ours	51.2	33.2
Ours + Iterative	119.4	71.4

Table 4: Comparison of the decoding time (s).

Efficiency Study Unlike Template-based BART, querying every n-gram span in the source sentence with each label with $O(n^2 * m)$ (where n is the length of source sentence and m is the size of the label set) time complexity, our proposed method queries labels in label set and directly generate slots with $O(n * m)$ time complexity at top. In theory, our method is much faster than Template-based BART, especially dealing with long sentences with sparse slots. To prove this, we conduct efficiency experiments by calculating the decoding time of each method on a TiTan XP GPU with batch size=8, and we set our max generation length at 40. As shown in Table 4, our method is about 8 times as fast as Template-based BART method, even more than 3 times as fast as theirs with Iterative Prediction Strategy. It is worth pointing that most slots are short and sparse in a sentence, which means our average generation length is short and with careful controlling when to end decoding, the time complexity of our method can be very close to the lower boundary $o(n)$.

5 Related Work

Prompt-based learning Prompt-based learning approaches have been a broadly discussed topic since large language models like GPT models (Brown et al., 2020) are hard to fine-tune in low-resource scenarios. Schick and Schütze (2021a,b) introduce manually prompts to text classification tasks. For natural language understanding (NLU) tasks, automatically searching discrete prompts methods are proposed such as Jiang et al. (2020); Shin et al. (2020); Gao et al. (2021). Meanwhile, due to the continuity of parameters in neural networks, continuous prompts for both text classification and generation tasks (Li and Liang, 2021; Liu et al., 2021b; Han et al., 2021) have been proposed. Unlike sentence-level tasks, prompting method is very complicated for slot tagging and NER tasks. Cui et al. (2021) proposes a template-based method querying every slot span with each label which is expensive for decoding. Different from them, we introduce an inverse paradigm for prompting slot tagging task. Note that inverse prompting (Zou et al., 2021) has a similar name to our work but is entirely different in method and task. They aim to generate prompt templates inversely. Amendable generation (Tian et al., 2021) share a similar idea of using Iterative Prediction Strategy to generate and revise dialog state. By contrast, we focus on a different task sequence labeling and first to introduce an Iterative Prediction Strategy to prompting models. There are also generation-based methods for sequence labeling (Yan et al., 2021), which is not a prompting method, since it re-initializes decoding layers and learns a generative model from scratch.

Few-shot slot tagging Previous few-shot slot tagging methods focus on metric learning based methods, which classify tokens with word-label similarity (Snell et al., 2017; Vinyals et al., 2016). Hou et al. (2020) leverage label name semantics to get better label representation and model label dependency in few-shot setting. Yang and Katiyar (2020) uses make a prediction based on the nearest neighbor sample instead of the nearest label representation. Besides, some works also explore training a model with additional data from non-slot-tagging task (Huang et al., 2020; Henderson and Vulic, 2021). Different from directly learning the few-shot slot tagging model, some researches explore to reformulate the slot tagging into other NLP tasks, Ma et al. (2021a) reforms slot tagging into a reading comprehension task, Yu et al. (2021) treats slot tagging as a retrieval task, Coope et al. (2020) uses span extracting task to extract slot and predict corresponding label and Cui et al. (2021) leverages prompts for few-shot NER. Different from those methods above, we are the first to reformulate slot tagging task into a prompt-based generation task.

6 Conclusion

In this paper, to liberate the prompting methods from burdensome prediction of slot-tagging tasks, we introduce a novel inverse prediction manner to prompting methods of slot-tagging, which significantly improves both the efficiency and accuracy. To further improve performance, we propose an Iterative Prediction Strategy for learning, which enable the prompting model to consider dependency between labels and refine prediction. Extensive experiments verify the effectiveness of the proposed method in various few-shot settings, indicating inverse prediction is a better fit for prompting of slot tagging task.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Sam Coope, Tyler Farghly, Daniela Gerz, Ivan Vulic, and Matthew Henderson. 2020. Span-convert: Few-shot span extraction for dialog with pretrained conversational representations. In *Proc. of the ACL*, pages 107–121.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online. Association for Computational Linguistics.

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

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. 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)*, pages 3816–3830, Online. Association for Computational Linguistics.

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

Matthew Henderson and Ivan Vulic. 2021. Convex: Data-efficient and few-shot slot labeling. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 3375–3389.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *Proc. of ACL*, pages 1381–1393. Association for Computational Linguistics.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2020. Few-shot named entity recognition: A comprehensive study. *arXiv preprint arXiv:2012.14978*.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. 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*, pages 7871–7880, Online. Association for Computational Linguistics.

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

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Jingjing Liu, Panupong Pasupat, Yining Wang, Scott Cyphers, and Jim Glass. 2013. Query understanding enhanced by hierarchical parsing structures. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 72–77. IEEE.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT understands, too. *CoRR*, abs/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*.

Jianqiang Ma, Zeyu Yan, Chang Li, and Yang Zhang. 2021a. Frustratingly simple few-shot slot tagging. In *Findings of the ACL*, pages 1028–1033.

Ruotian Ma, Xin Zhou, Tao Gui, Yiding Tan, Qi Zhang, and Xuanjing Huang. 2021b. Template-free prompt tuning for few-shot NER. *CoRR*, abs/2109.13532.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational LinguisticsACL*, pages 1064–1074. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language ProcessingEMNLP*, pages 1532–1543.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Timo Schick and Hinrich Schütze. 2021a. 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*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. 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*, pages 2339–2352, Online. Association for Computational Linguistics.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45(11):2673–2681.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Xin Tian, Liankai Huang, Yingzhan Lin, Siqi Bao, Huang He, Yunyi Yang, Hua Wu, Fan Wang, and Shuqi Sun. 2021. Amendable generation for dialogue state tracking. *CoRR*, abs/2110.15659.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing SystemsNIPS*, pages 3630–3638.

Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3):63:1–63:34.

Sam Wiseman and Karl Stratos. 2019. Label-agnostic sequence labeling by copying nearest neighbors. In *Proceedings of the 57th Annual Meeting of the Association for Computational LinguisticsACL*, pages 5363–5369.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *Proc. of the ACL/IJCNLP*, pages 5808–5822. Association for Computational Linguistics.

Yi Yang and Arzoo Katiyar. 2020. 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)*, pages 6365–6375, Online. Association for Computational Linguistics.

Dian Yu, Luheng He, Yuan Zhang, Xinya Du, Panupong Pasupat, and Qi Li. 2021. Few-shot intent classification and slot filling with retrieved examples. In *Proc. of the NAACL*, pages 734–749.

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*.

Morteza Ziyadi, Yuting Sun, Abhishek Goswami, Jade Huang, and Weizhu Chen. 2020. Example-based named entity recognition. *CoRR*, abs/2008.10570.

Xu Zou, Da Yin, Qingyang Zhong, Hongxia Yang, Zhilin Yang, and Jie Tang. 2021. Controllable generation from pre-trained language models via inverse prompting. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 2450–2460. ACM.