Zero-Shot Slot Filling with Slot-Prefix Prompting and Attention Relationship Descriptor

Qiaoyang Luo, Lingqiao Liu*

The University of Adelaide
{qiaoyang.luo, lingqiao.liu}@adelaide.edu.au

Abstract

This paper addresses zero-shot slot filling, which tries to build a system that can generalize to unseen slot types without any training data. The key to zero-shot slot-filling is to match the tokens from the utterance with the semantic definition of the slot without training data in the target domain. This paper tackles this problem by devising a scheme to fully leverage pre-trained language models (PLMs). To this end, we propose a new prompting scheme that utilizes both learnable tokens and slot names to guide the model to focus on the relevant text spans for a given slot. Furthermore, we use attention values between tokens to form a feature descriptor for each token, which is motivated by the fact that the attention value in a PLM naturally characterizes various relationships, e.g., syntactic or semantic, between tokens. By further consolidating those features with an additional transformer-based aggregation module, we create a simple-but-effective zero-shot slot filling system that can achieve significantly better performance than the previous methods, as demonstrated by our experimental studies.

Introduction

Slot filling is a task to extract values of certain types of attributes and plays an essential role in task-oriented dialogue systems. In the past years, supervised-learning-based slot filling systems have achieved great success (Wang, Shen, and Jin 2018; Niu et al. 2019; Qin et al. 2019; Wu et al. 2020). However, those methods often require substantial amount of annotations to train the model, which becomes a significant limitation in real-life scenarios when we adapt the model to a new domain, where the training data might be scarce or expensive to obtain. Zero-shot slot filling tries to overcome this limitation by learning a slot filling system that can generalize to unseen slot types and domains without any training data for the unseen slots.

In such a setting, the only information about a new slot type is given by the semantic description of a slot, dubbed the **slot description**, either in the form of slot type name, textual definition, or exemplar words. The key challenge of the task is to built a system that can identify the span that fits the slot description, which should be generalizable to unseen domains.

*Corresponding Author.
Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Recent works (Liu et al. 2020; He et al. 2020; Wang et al. 2021; Du et al. 2021; Yu et al. 2021) take different formulations to model the zero-shot slot-filling problem. For example, the works in (Liu et al. 2020; He et al. 2020; Wang et al. 2021) treat zero-shot slot filling as a two-stage sequence labeling task. In the first stage, they perform BIO classification to identify potential slot spans. Then, they use another encoder to encode slot descriptions and classify the slot type by comparing each slot span and slot descriptions. State-of-the-art methods (Du et al. 2021; Yu et al. 2021) formulate the zero-shot slot-filling task as a single-stage question-answering (QA) problem to take advantage of the build-in matching capability of a pretrained language model (PLM). In such a formulation, slot description, which is formulated as a question, and the utterance are processed by a single encoder, i.e., the PLM. Thus, the tokens from the utterance and the slot description can interact with each other across all layers of a PLM.

In this paper, we tackle zero-shot slot-filling by devising a novel scheme to fully leverage a PLM. To this end, we formulate zero-shot slot filling as a special sequence tagging problem guided by a prompt created from slot descriptions. The hybrid prompt consists of both the tokens in the original slot descriptions (discrete prompt) and the learnable tokens (continuous prompt) and is dubbed **Slot-Prefix (SP)** prompting since the learnable tokens are appended as a prefix to the slot-description tokens, as shown in Figure 1. More specifically, the tokens from slot descriptions will be compared against the utterance tokens through the attention modules across all layers in a PLM and thus can leverage the token matching capability pre-acquired at the pre-training stage of a PLM. Additionally, concatenating all slot descriptions in the discrete prompt allows the model to explore relationships among the slot descriptions, whereas QA-based methods ignore the relationships by processing each slot type independently and may cause overlap prediction of different slot types during inference. As part of SP prompting, the continuous prompt is designed to strengthen the PLM understanding of contextual meanings associated with slot descriptions and to learn generalization capabilities to enable more accurate span localization across domains.

In addition to the slot-prefix prompting scheme, we also utilize self-attention values between a pair of tokens to characterize their relationship and accumulate attention val-

ues from all heads and layers of a PLM to form a feature representation called **Attention Relationship Descriptor (ARD)**, for each utterance token. The motivation of this scheme is that the attention value in a PLM naturally characterizes various relationships, e.g., syntactic or semantic, between tokens (Vig and Belinkov 2019; Clark et al. 2019). This build-in relationship representations could provide valuable prior for zero-shot slot-filling tasks. Finally, we employ a transformer (Vaswani et al. 2017) to aggregate ARDs from different query tokens to produce the final prediction.

Through our experimental study, we demonstrate that our proposed method achieves superior performance over state-of-the-art methods in both SNIPS and TOP datasets. Our contributions can be summarized as follows:

- We formulate the zero-shot slot-filling task as a prompt-guided sequence tagging problem and propose a novel hybrid prompting scheme - SP prompting - which effectively combines learnable prompts and slot description prompts to solve this problem. Our proposed prompting scheme has the potential to be adapted to other sequence tagging systems, such as Named-entity recognition.
- We introduce the attention relationship descriptor and explore various aggregation modules to fully leverage a pre-trained language model for zero-shot learning slot-filling. This approach can be applied to other zero-shot/few-shot NLP tasks.

Related Work

Zero-Shot Slot Filling

Zero-shot slot filling has received much attention in recent years due to its potential in alleviating the burden of collecting annotations. As an early attempt in this direction, Bapna et al. (2017) tackled the problem by augmenting utterance token embeddings with the embedding of the slot descriptions of a target slot type and produced BIO tags to indicate the target slot span. Following the framework of Bapna et al. (2017), Shah et al. (2019) further explored both slot descriptions and slot examples to improve the performance. Liu et al. (2020) proposed a coarse-to-fine approach to solving the problem in two stages. They firstly identify if the tokens are slot entities or not and then generate the final prediction by measuring the representation similarities between the utterance and slot descriptions. He et al. (2020); Wang et al. (2021) incorporated constrastive learning strategies to improve the robustness and generalization of the zero-shot slot filling systems. More recently, Du et al. (2021); Yu et al. (2021) proposed QA-based approaches and leveraged the PLMs to achieve the state-of-the-art performance in zero-shot slot filling.

Prompting

Prompt is a token, sentence, or template which reformulates the input of an NLP task to better explore the intrinsic knowledge of PLMs, and it achieved success in various NLP tasks (Radford et al. 2019; Petroni et al. 2019; Cui et al. 2021; Schick and Schütze 2021). Brown et al. (2020) firstly proposed the idea of prompting. It demonstrates impressive performance on many NLP tasks by using few samples as a prompt, even without updating the parameters of the PLM. However, an inappropriate prompt can cause sub-optimal performance (Jiang et al. 2020) and manually designing a good prompt is time-consuming for specific tasks. Gao, Fisch, and Chen (2021); Shin et al. (2020) proposed to generate prompts automatically and achieved better performance than manually designed prompts in few-shot learning. These automatically generated prompts are represented in discrete space, and they are constrained to be human-interpretable natural language (Liu et al. 2021a). Compared with discrete prompts, Liu et al. (2021c); Lester, Al-Rfou, and Constant (2021); Qin and Eisner (2021) proposed trainable continuous prompts in the embedding space of PLMs, and these Prompt tuning methods freeze the model and only train the continuous prompts. In this paper, our methods can be regarded as hard-soft hybrid prompts consisting of continuous learnable prompts for BIO classification and hard prompts for slot descriptions. In addition, our proposed SP prompt is verified to be effective on both finetuning and prompt tuning in zero-shot slot filling tasks.

Use of Attention Features

Vig and Belinkov (2019) analyzed the structure of attention in GPT-2 (Radford et al. 2019) and found that the scores from many attention heads have correspondences with part-of-speech (POS) tags and syntactic dependency relations. Few works attempted to leverage the rich linguistic knowledge encoded in the attention heads of PLMs, Kim et al. (2020); Li et al. (2020) explored synatatic information in attention features for unsupervised consistency parsing. Clark et al. (2019) integrated the attention features of BERT with Glove word embedding and obtained decent performance on dependency parsing. Tikhonov and Ryabinin (2021) discussed several different operations of using attention scores for commonsense reasoning. Unlike the previous methods explore the attention features in a shallow manner, we proposed a transformer-based aggregation schema to fully explore the attention features in zero-shot slot filling tasks, which is proved to be crucial for good performance.

Methodology

Figure 1 gives an overview of the proposed method. The input utterance is appended by the Slot-Prefix prompt that consists of the slot description and three types of learnable tokens: [B], [I], [O], corresponding to the three labels in traditional BIO encoding. The prompt-appended utterance is fed into a PLM, e.g., BERT, to extract attention relationship descriptors. Since ARD is defined for each pair of tokens, for an input sequence with n tokens, there will be $n \times n$ pairs of ARDs. If we organize them as an array, then each row can be viewed as a vector sequence sharing the same "query token". Our method uses an ARD aggregation module to aggregate those sequences to the final prediction. In the following, we will first formally define the problem and notations, and then we will elaborate on the Slot-Prefix prompting scheme, the design of the attention relationship descriptor, the ARD aggregation module, and the training process sequentially.

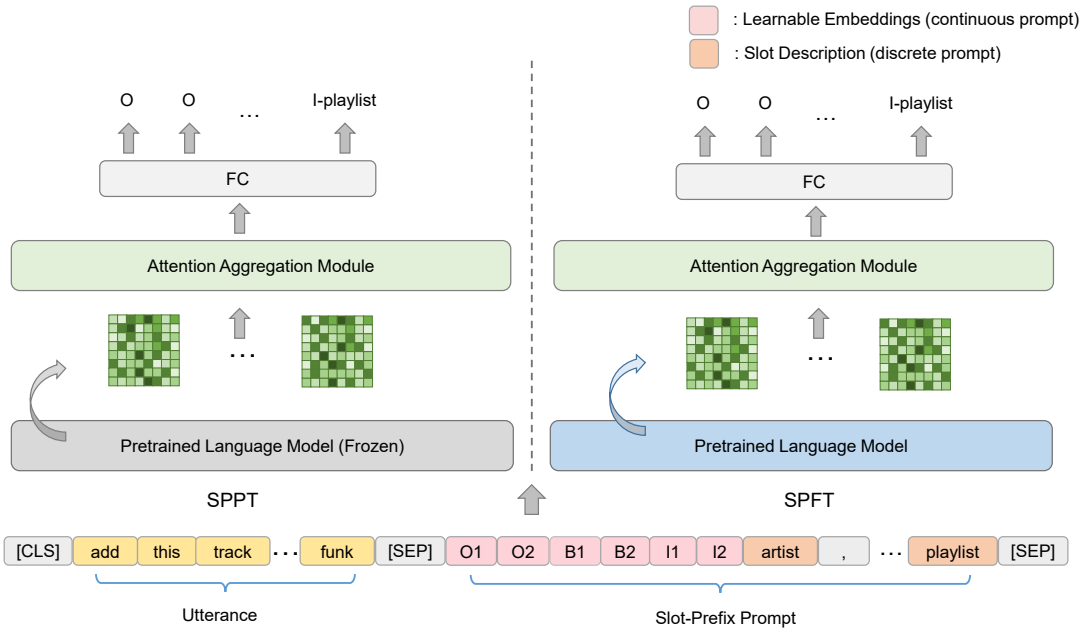

Figure 1: Overview of the proposed zero-shot slot filling algorithm. An utterance and all the slot descriptions are converted into a sequence by following the Slot-Prefix format. Then a PLM is used to extract ARDs from it. The extracted ARDs are further transformed by the ARD aggregation module into the prediction of tags.

Problem Formulation and Notations

In zero-shot slot filling tasks, we follow the convention of using BIO encoding schema to convert slot filling into a sequence labeling problem. In other words, a slot type will be appended "B" or "I" to indicate the start or inside of a slot span. "O" represents other tokens that do not belong to any slot types. For example, the slot type "artist" will be converted to two tags - "B-artist" and "I-artist." Therefore, such an encoding scheme will create $2l + 1$ tags for l slot types. A zero-shot slot filling system will compare each token in the utterance against each slot description and predict if it locates at the start, inside or outside of the compared slot type. Without loss of generality, we use $S_c = \{s_1^c, \cdots, s_{n_c}^c\}$ to denote the semantic description of a slot type c, where n_c denotes the number of tokens in its description. Our model aims to learn a map function that can produce the tagging sequence $Y = \{y_1, y_2, \cdots, y_n\}$, $y_i \in \{1, \cdots, 2l + 1\}$ based on the input utterance $X = \{x_1, x_2, \cdots, x_n\}$ and a set of slot types and their slot descriptions $\mathcal{S} = \{S_1, \cdots, S_l\}$.

Slot-Prefix Prompting

To fully leverage a PLM for zero-shot slot filling, we convert all the available information into an input sequence. In particular, we append a prompt to the input utterance which creates the input sequence following the format: "[CLS] Utterance [SEP] $[O_1][O_2]...[O_K]$ $[B_1][B_2]...[B_K]$ $[I_1][I_2]...[I_K]$ Slot description 1 [,] Slot description 2 [,] ... [SEP]". In the above format, the embeddings for $[B_1]...[B_K]$, $[I_1]...[I_K]$, $[O_1]...[O_K]$ are learnable during training (learnable tokens)[1]. Among them, O-type tokens are used to denote "does not belong to any slot type"; B-type tokens and I-type tokens act as prefix tokens to the slot description to indicate "the beginning of a slot type" and "within a slot type", respectively. We use K tokens for each type of learnable tokens. Such a scheme is akin to recently proposed prompt tuning approaches (Liu et al. 2021a; Sun et al. 2021; Li and Liang 2021). Figure 1 gives an example of how to encode an utterance into the input sequence by following the Slot-Prefix format.

The use of [B],[I],[O]-type tokens could compensate the original slot description tokens to drive the PLM to produce more accurate slot span localization. Note that we use the same set of [B], [I], [O]-type tokens for all slot types. In other words, those learnable tokens are slot-type agnostic, which resonates the design of two-stage approach: the first stage detects slot-type-agnostic span. Also, note that Slot-Prefix prompting could process multiple slot types in one pass. This is in sharp contrast to existing QA-based approaches (Du et al. 2021; Yu et al. 2021) which can only process with one slot type in a single pass of the PLM.

In our study, we experiment with two ways of training the learnable token. The first one is to keep the PLM fixed but only train the learnable token and ARD aggregation model (similar to the recently proposed prompt-tuning (Li and Liang 2021; Liu et al. 2021b)), and the other is to also

[1]In our implementation, we use deep prompting (Liu et al. 2021b), which adds learnable tokens to each layer of the PLM. This scheme usually leads to better performance. For the convenience of discussion, we omit this implementation details in the following discussion.

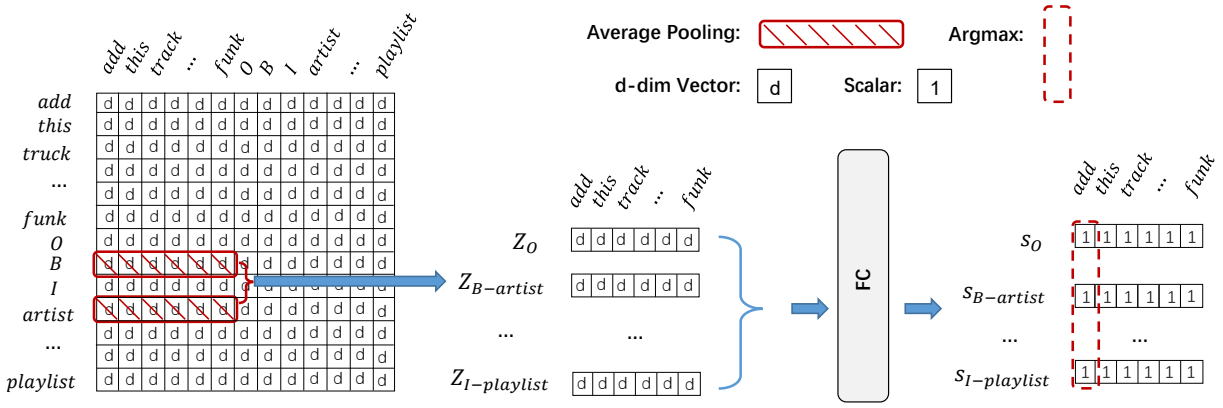

Figure 2: The illustration of the process of converting ARDs into the final prediction. Please refer to the last paragraph of section for more details.

fine-tune the PLM. It turns out that prompt tuning works better, and we postulate that prompt tuning keeps the parameters of a PLM intact during training and thus maintains knowledge learned from pre-training. Such a scheme could prevent the model from overfitting to the limited number of domains in the training set.

For the convenience of discussion, we denote the input sequence as $\{u_1, u_2, \cdots, u_n, p_1, p_2, \cdots, p_m\}$, where u_i denotes the utterance token, p_i denotes the prompt tokens including continuous learnable prompts and discrete prompts. We further use $\mathcal{O}$,$\mathcal{B}$,$\mathcal{I}$, and $\mathcal{S}_k$ denote index set corresponding to the O,B,I tokens and tokens from the k-th slot description. Then we define $2l+1$ tags to indicate the B or I version of the slot-types and the O tag. We then create $2l+1$ index sets to represent relevant tokens for each tag:

$$\mathcal{T}_y = \begin{cases} \mathcal{B} \bigcup \mathcal{S}_y, & \text{if } y \leq l \\ \mathcal{I} \bigcup \mathcal{S}_{y-l}, & \text{if } l+1 \leq y \leq 2l \\ \mathcal{O}, & \text{if } y = 2l+1. \end{cases} \quad (1)$$

In other words, for the B(I)-version of the c-th slot type, the B(I) tokens and the tokens from the c-th slot type will be considered relevant. The O tag is totally slot-type agnostic.

Attention Relationship Descriptor Extraction

As mentioned before, our method utilizes the attention values of a Transformer-based PLM to form feature descriptors to characterize the relationship between tokens. Formally, let's denote that the self-attention value calculated at the l-th layer and h-th attention head in the multi-head attention module (MHA) as $\mathbf{A}^{h,l} \in \mathrm{R}^{n \times n}$, where n is the length of the input sequence. Each (i, j) element of $\mathbf{A}^{h,l}$ depicts the relatedness of two tokens in the view of the l-th layer, h-th MHA module.

For a pair of tokens, we can collect attention values from different MHA modules and layers. The collection of those attention values will produce a vector:

$$\mathbf{a}_{i,j} = [A^{1,1}_{i,j}, \cdots, A^{1,H}_{i,j}, \cdots, A^{L,1}_{i,j}, \cdots, A^{L,H}_{i,j}] \in \mathrm{R}^d,$$

where L denotes the total number of layers, H denotes the number of MHAs per layer. For example, in a $\text{BERT}_{\text{BASE}}$ model, $L = 12$ and $H = 12$, and the resulted $\mathbf{a}_{i,j}$ is a 144-dimensional vector. $\mathbf{a}_{i,j}$ represents a collection of view on how the token i and j is related from multiple layers and multiple heads. It naturally depicts the relationship between those two tokens. In this work, we call $\mathbf{a}_{i,j}$ attention relationship descriptor or ARD in short.

Note that $\mathbf{a}_{i,j} \neq \mathbf{a}_{j,i}$ since the self-attention in the Transformer is asymmetric. The difference between $\mathbf{a}_{i,j}$ and $\mathbf{a}_{j,i}$ is which token, i or j, is used as the query in the self-attention calculation. In the following, we assume the first index of $\mathbf{a}_{i,j}$, i.e. i, corresponding to query and call it "query token".

Attention Relationship Descriptor Aggregation

For an input sequence $\{u_1, u_2, \cdots, u_n, p_1, p_2, \cdots, p_m\}$, we can extract $(n+m)^2$ ARDs. In other words, for each "query" token indexed by i, we can extract an ARD sequence with $n+m$ ARD vectors characterizing the relationship between each token to the query token. Formally we could represent those sequences as:

$$\mathbf{Ar}_i = [\mathbf{a}_{i,1}, \mathbf{a}_{i,2}, \cdots, \mathbf{a}_{i,(n+m)}] \in \mathrm{R}^{(n+m) \times d}.$$

To aggregate ARD features, we propose to use a shared transformer f to process $\mathbf{Ar}_i$ [2], that is,

$$\mathbf{Z}_i = [\mathbf{z}^1_i, \mathbf{z}^2_i, \cdots, \mathbf{z}^{n+m}_i] = f(\mathbf{Ar}_i) \in \mathrm{R}^{(n+m) \times d}, \quad (2)$$

where the use of f could capture the temporal patterns of the ARD sequences and it converts each ARD feature in the sequence into a d-dimensional representation. Finally, we aggregate the ARD sequences corresponding to the "query token" from the y-th tag set $\mathcal{T}_y$

$$\bar{\mathbf{Z}}_y = [\bar{\mathbf{z}}^1_y, \bar{\mathbf{z}}^2_y, \cdots, \bar{\mathbf{z}}^n_y] = \frac{1}{|\mathcal{T}_y|}[\sum_{i \in \mathcal{T}_y} \mathbf{z}^1_i, \sum_{i \in \mathcal{T}_y} \mathbf{z}^2_i, \cdots, \sum_{i \in \mathcal{T}_y} \mathbf{z}^n_i] \in \mathrm{R}^{n \times d}.$$

[2] In our implementation, we apply the transformer along each row of the ARDs. We can also apply the transformer along each column of the ARDs. However, we empirically find those two schemes produce similar performance.

Finally, we apply a fully-connected layer g to convert $\bar{\mathbf{z}}_y^j \in \mathrm{R}^d$ into a matching value and select the class corresponding to the highest matching score as the final prediction:

$$\hat{y}^j = \underset{c}{argmax}\, g(\mathbf{z}_c^j). \tag{3}$$

The above process can also be explained graphically as shown in Figure 2. We first extract the ARD for each pair of tokens, and this creates a 2-D array of ARD features. Then we apply a shared transformer along each row of the array. Then we find rows that correspond to the slot description of the y-th tag and perform average pooling along the column direction. Then a fully-connected (FC) layer converts the pooled vectors into matching scores.

Model Training

We train our model from a training set with provided utterance and slot description pair. In our experiments, we simply use class names as slot descriptions to encourage fair comparison with existing approaches. To calculate the training loss, we first convert the matching score after the FC layer into a probability on the tag class in Eq. 3 into a probability by $Pr(y_j = y) = \frac{\exp(g(\mathbf{z}_y^j))}{\sum_k \exp(g(\mathbf{z}_k^j))}$. Then we can use cross-entropy loss to encourage $Pr(y_j = y)$ producing the highest probability for the ground-truth tag class.

Experiments

Datasets

We evaluate the proposed method on SNIPS (Coucke et al. 2018) which is a public spoken language understanding dataset. It includes 39 distinct slots types from 7 domains, and each domain contains around 2000 samples. Following the previous zero-shot evaluation setup from (Liu et al. 2020), we select one domain as the target domain each time while using the other six domains as the source domain. We also evaluated our model at TOP dataset (Gupta et al. 2018) that is task-oriented semantic parsing dataset with 36 slot types. We follow the settings from (Du et al. 2021) to use all seven domains of SNIPS as training data.

Comparing Methods

We compare our framework with the following baselines:

Concept Tagger (CT) Bapna et al. 2017 solved the problem by incorporating the embedding of slot name and descriptions with utterance.

Robust Zero-shot Tagger (RZT) Based on CT, Shah et al. 2019 leveraged both slot descriptions and few examples of slot values to improve the performance.

Coarse-to-fine Approach (Coach) Liu et al. 2020 proposed two-stage frameworks to perform BIO-classification first and slot type classification based on BIO results. They further boosted the performance by introducing template regularization to regularize the utterance representation.

Contrastive Zero-Shot Learning with Adversarial Attack (CZSL) Following Coach framework, He et al. 2020 improved the performance by contrastive loss which maps utterance representation to the corresponding slot description representations. Furthermore, adversarial attack strategy is applied to improve model robustness.

Zero-Shot BERT Tagger (ZSBT) Du et al. 2021 investigated the performance of utilizing BERT (Devlin et al. 2018) to predict BIO classes for each token with encoding slot descriptions and types individually.

Prototypical Contrastive learning and Label Confusion strategies (PCLC) Wang et al. 2021 proposed a method to dynamically refine slot prototypes' representations and improved the performance based on Coach framework.

QA-driven slot filling framework (QASF) Contrary to previous methods, Du et al. 2021 introduced QA-based framework and leveraged the PLMs to solve the problem.

Our method is denoted as **SPFT** and **SPPT**, with the former indicating the version of fine-tuning the PLM and the latter indicating the version of just tuning the learnable tokens in the prompt.

Results and Discussion

Main Results

Table 1 and Table 2 shows the main results of our method. As seen, our finetuning-based (SPFT) and prompt tuning-based (SPPT) methods achieve superior performance on six domains and final average F1 scores under zero-shot slot filling tasks. Comparing the two-stage methods (Coach, CZSL and PCLC) based on adopting Conditional Random Field (CRF) for BIO classification, SPFT and SPPT surpass them by a large margin in final average F1 scores. Comparing with the single-stage methods (CT, RZT, ZSBT and QASF), our methods also requires a single stage process but consistently outperforms these methods on both SNIPS and TOP datasets. For example, on TOP dataset, SPFT and SPPT outperforms ZSBT by over 11% and 13%. Note that both ZSBT and our methods leverage the pre-trained language models e.g. BERT. QASF achieves previous state-of-the-art performance by transforming the sequence labeling tasks into question-answering tasks and using BERT as base model to encode question and utterance for each slot type, which is most similar to our approach. By comparing against QASF, we can observe that SPFT and SPPT can outperform QASF on SNIPS by 4-6%, on TOP by 9%. This validates our claim that our model can fully leverage a PLM. Finally, we observe that SPPT can be comparable to SPFT or even outperform SPFT despite SPPT uses much fewer learnable parameters. It seems that using smaller number of learnable parameters can be beneficial to generalize from limited training domains to unseen domains.

Ablation Studies

In this section, we conduct several ablation study to thoroughly investigate various factors to lead better performance. 1. The importance of using the ARD. 2. The design of the ARD aggregation module. 3. The impact of the number of learnable tokens. 4. The use of other PLMs.

Study 1: the importance of the ARD representation Our approach uses an unconventional attention relation descriptor to represent each token. To investigate its necessity,

Training Setting	SNIPS Zero-Shot							
Model ↓	ATP	BR	GW	PM	RB	SCW	FSE	**Avg F1**
CT (Bapna et al. 2017)	38.82	27.54	46.45	32.86	14.54	39.79	13.83	30.55
RZT (Shah et al. 2019)	42.77	30.68	50.28	33.12	16.43	44.45	12.25	32.85
Coach (Liu et al. 2020)	50.90	34.01	50.47	32.01	22.06	46.65	25.63	37.39
CZSL (He et al. 2020)	53.89	34.06	52.24	34.59	31.53	50.61	30.05	40.99
ZSBT$^{\text{BERT}}$ $*$	55.78	49.34	56.58	28.35	27.09	57.61	20.05	42.18
PCLC (Wang et al. 2021)	59.24	41.36	54.21	34.95	29.31	53.51	27.17	42.82
QASF (Du et al. 2021)	59.29	43.13	59.02	33.62	33.34	59.90	22.83	44.45
SPFT (ours)	58.16	44.94	66.06	36.53	28.02	67.77	34.54	48.00
SPPT (ours)	55.61	44.39	65.53	41.19	33.87	71.95	39.01	50.22

Table 1: Zero-shot Slot Filling F1-scores (%) for seven different target domains on SNIPS. SPFT refers to our proposed SP prompting based on finetuning, SPPT refers to SP prompting based deep prompt tuning.

Model	backbone	Top Zero-Shot
ZSBT	BERT	8.82
QASF	BERT	10.27
SPFT	BERT	19.55
SPPT	BERT	**19.78**

Table 2: Results of zero-shot slot filling on TOP dataset.

Model	TOP	SNIPS
SPFT-EBD	11.13	46.08
SPFT	19.55	48.00
SPPT-EBD	18.70	45.24
SPPT	**19.78**	**50.22**

Table 3: The importance of using attention features. SPFT-EMD and SPPT-EMD are variants of SPFT and SPPT that use embedding features instead.

we compare our method with a variant without using attention feature. Formally, we could represent the last layer embedding of a PLM as $\{\mathbf{e}_1^t, \mathbf{e}_2^t, \cdots, \mathbf{e}_n^t, \mathbf{e}_1^p, \mathbf{e}_2^p, \cdots, \mathbf{e}_m^p\}$, where $\mathbf{e}_i^t$ and $\mathbf{e}_i^p$ indicate the embedding from the utterance and from the prompt, respectively. Using the index system in Eq. 1, we can calculate the matching score between the j-th token and the y-th tag as $\mathbf{e}_j^{t\top}\left(\sum_{i\in\mathcal{I}_y} e_i^p\right)$.

To make a fair comparison, we apply a transformer with the same amount of parameters to process the last-layer embedding. The performance comparison is shown in Table 3. As seen, this alternative is still inferior to our approach, in both fine-tuning and prompt-tuning settings. Despite its relatively lower performance, this variant performs surprisingly well in comparison with other competitive approaches including QASF, which also uses a PLM. This observation verify that both Slot-prefix Prompting and the attention features contribute to the good performance of our method.

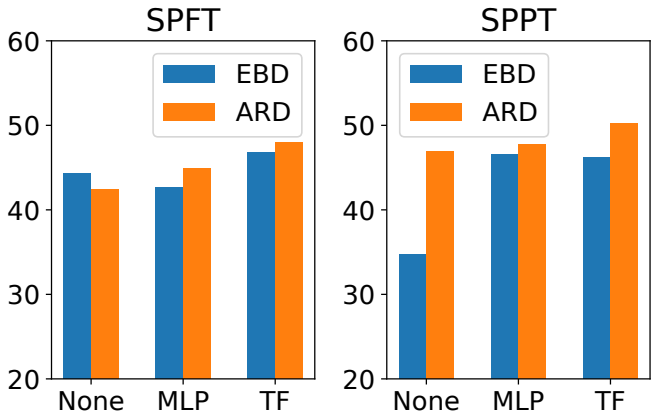

Figure 3: F1-scores of our proposed SP models based on different features and attention aggregation modules .

Study 2: the design of ARD aggregation module After extracting the ARDs, our method does not directly use them for prediction but feeds them into the ARD aggregation module as in Eq. 2. In our implementation, the ARD is aggregated via a shallow transformer encoder. One may wonder (1) if using the extra aggregation module is necessary and (2) if using a transformer is necessary. To address those two concerns, we designed the following variants of our method. Note that the ARD aggregation module does not change the size of the input (still with the size of $(n+m)\times d$). Variation 1 (ARD-Only) does not use any aggregation module, that is, $\mathbf{z}_i = \mathbf{z}_i$; variation 2 (ARD-MLP) uses a multilayer perceptron (MLP) to replace the transformer in Eq. 2. Conceptually, the major difference between ARD-MLP and ARD-TF is that the former only aggregates ARDs corresponding to the current token while the latter also consider ARDs from the other tokens. We also test the those variant with SPFT-EMD and SPPT-EMD. The results are shown in Figure 3.

As seen for ARD-based approaches, without using any

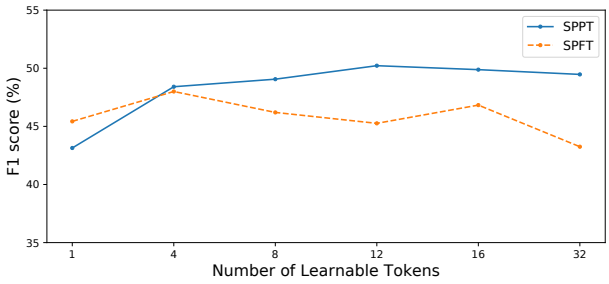

Figure 4: F1-scores of our proposed SP models based on different features and model structuresc .

aggregation module, the average F1 drops 3.2% for SPPT and 5.6% for SPFT. Using an MLP can further boost the average F1 to 0.8% for SPFT and 2.5% for SPPT. However, there is still around 2% gap when compared with the transformer-based aggregation module. This provides strong evidence to support the use of a transformer for ARD aggregation. Another interesting observation is that even without using the transformer-based aggregation module, our SPPT-based method still achieves superior performance than the other approaches, e.g., QASF. This also illustrates the advantage of the ARD representation. For embedding-based approach, we can see that using MLP or TF can still boost the performance. However, depending on whether fine-tuning the model or perform prompt tuning, the improvement varies. The best performance attained when using MLP to process embedding. This leads to around 1% improvement over SPPT-EMD with transformer. However, this result is still inferior to SPPT with ARDs.

Study 3: the impact of the number of learnable tokens The special learnable token [B], [I] and [O] can be implemented with one or many learnable embeddings. It is thus of interest to investigate the impact of the number of learnable tokens. In Figure 4, we conduct an experiment to test the performance by vary the number of learnable [B], [I], [O] tokens. Generally, it seems that the number of learnable tokens have larger impact on SPPT. In such a case, using more than 4 learnable tokens usually produces better performance. The performance of SPFT does not significantly vary with the number of learnable tokens. This observation is reasonable since SPFT has more parameters to tune than that in SPPT.

Study 4: using other PLMs Our existing results are based on the BERT model. We here investigate the impact of changing PLM. In Table 4, we show the results by replacing BERT with RoBERTa. As seen, the conclusion remains the same: (1) SPPT achieves overall best performance and prompt tuning seems to be more effective than fine-tuning (2) Using ARD is better than using embedding. (3) SPPT still outperforms the existing approaches, but interestingly using RoBERTa does not bring too much performance boost over BERT as we initially expected.

	SPFT		SPPT	
Features	EBD+TF	ARD+TF	EBD+TF	ARD+TF
Zero-Shot	45.23	**47.53**	43.18	**49.95**

Table 4: Results zero-shot slot filling on SNIPS dataset of our model using Roberta as backbone .

Error Analysis and Limitations

We analyze the error pattern of our approach and also the SPPT-EBD variant. We find that SPPT (with ARD or with EBD) model often makes mistakes when classifying slot type, in particular when the utterance contains similar words in the slot description. For example, for the utterance “add the artist verano to my michael mantler playlis”, the token “artist” is labeled as “B-music-item” but was incorrectly classified as “B-artist” in our method. For SPPT(with ARD) and SPPT-EBD, we found that using ARD (SPPT) achieves better span localization and slot type classification results.

Although our methods achieve SOTA performance on both SNIPS and TOP dataset, several key reasons limit the performance of our models on zero-shot slot-filling tasks. (1) Some domains contain fine-grained slot types which cause ambiguity to the model to distinguish the minor difference among them in utterances. For example, the fine-grained slot types in BookRestaurant(BR) includes ”spatial_relation”, ”poi”, ”city” and ”country” and TOP dataset includes ”attendee_event”, ”attribute_event”, ”category_event” and ”organizer_event”. (2) The model focuses on the lexical meanings of tokens rather than the context in some cases. For example, ”book a reservation for city tavern in long bridge”, the truth label for the token ”city” is ”B-restaurant_name” but the model prediction is ”B-poi”. (3) Using slot names only constraints the model to understand the exact meaning of label semantics. For example, ”B-party_size_description” in BookRestaurant(BR) in majority cases refers to the names of participants in a party.

Conclusion

In this paper, we have presented a novel approach for zero-shot slot filling based on prompt-guided sequence tagging and attention relationship descriptors. Our proposed method has demonstrated superior performance compared to state-of-the-art methods on both SNIPS and TOP datasets. Our contributions include formulating the slot filling problem as a prompt-guided sequence labeling task, proposing a novel hybrid prompting scheme, and leveraging attention relationship descriptors (ARD) and exploring various ARD aggregation modules to further improve the performance. Moreover, our approach is flexible and can be extended to other sequence tagging problems. One possible direction for future work is to investigate the effectiveness of our approach on other NLP tasks that require zero-shot/few-shot learning.

References

Bapna, A.; Tur, G.; Hakkani-Tur, D.; and Heck, L. 2017. Towards zero-shot frame semantic parsing for domain scaling. *arXiv preprint arXiv:1707.02363*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Clark, K.; Khandelwal, U.; Levy, O.; and Manning, C. D. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Coucke, A.; Saade, A.; Ball, A.; Bluche, T.; Caulier, A.; Leroy, D.; Doumouro, C.; Gisselbrecht, T.; Caltagirone, F.; Lavril, T.; et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Cui, L.; Wu, Y.; Liu, J.; Yang, S.; and Zhang, Y. 2021. Template-Based Named Entity Recognition Using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 1835–1845.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Du, X.; He, L.; Li, Q.; Yu, D.; Pasupat, P.; and Zhang, Y. 2021. QA-Driven Zero-shot Slot Filling with Weak Supervision Pretraining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 654–664.

Gao, T.; Fisch, A.; and Chen, D. 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)*, 3816–3830.

Gupta, S.; Shah, R.; Mohit, M.; Kumar, A.; and Lewis, M. 2018. Semantic Parsing for Task Oriented Dialog using Hierarchical Representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2787–2792.

He, K.; Zhang, J.; Yan, Y.; Xu, W.; Niu, C.; and Zhou, J. 2020. Contrastive Zero-Shot Learning for Cross-Domain Slot Filling with Adversarial Attack. In *Proceedings of the 28th International Conference on Computational Linguistics*, 1461–1467.

Jiang, Z.; Xu, F. F.; Araki, J.; and Neubig, G. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8: 423–438.

Kim, T.; Choi, J.; Edmiston, D.; and Lee, S.-g. 2020. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. *arXiv preprint arXiv:2002.00737*.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3045–3059.

Li, B.; Kim, T.; Amplayo, R. K.; and Keller, F. 2020. Heads-up! unsupervised constituency parsing via self-attention heads. *arXiv preprint arXiv:2010.09517*.

Li, X. L.; and Liang, P. 2021. 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)*, 4582–4597.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Liu, X.; Ji, K.; Fu, Y.; Du, Z.; Yang, Z.; and Tang, J. 2021b. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Liu, X.; Zheng, Y.; Du, Z.; Ding, M.; Qian, Y.; Yang, Z.; and Tang, J. 2021c. GPT understands, too. *arXiv preprint arXiv:2103.10385*.

Liu, Z.; Winata, G. I.; Xu, P.; and Fung, P. 2020. Coach: A coarse-to-fine approach for cross-domain slot filling. *arXiv preprint arXiv:2004.11727*.

Niu, P.; Chen, Z.; Song, M.; et al. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. *arXiv preprint arXiv:1907.00390*.

Petroni, F.; Rocktäschel, T.; Riedel, S.; Lewis, P.; Bakhtin, A.; Wu, Y.; and Miller, A. 2019. Language Models as Knowledge Bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2463–2473.

Qin, G.; and Eisner, J. 2021. 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*, 5203–5212.

Qin, L.; Che, W.; Li, Y.; Wen, H.; and Liu, T. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. *arXiv preprint arXiv:1909.02188*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Schick, T.; and Schütze, H. 2021. 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*, 255–269.

Shah, D. J.; Gupta, R.; Fayazi, A. A.; and Hakkani-Tur, D. 2019. Robust zero-shot cross-domain slot filling with example values. *arXiv preprint arXiv:1906.06870*.

Shin, T.; Razeghi, Y.; Logan IV, R. L.; Wallace, E.; and Singh, S. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4222–4235.

Sun, Y.; Zheng, Y.; Hao, C.; and Qiu, H. 2021. NSP-BERT: A Prompt-based Zero-Shot Learner Through an Original Pre-training Task–Next Sentence Prediction. *arXiv preprint arXiv:2109.03564*.

Tikhonov, A.; and Ryabinin, M. 2021. It's All in the Heads: Using Attention Heads as a Baseline for Cross-Lingual Transfer in Commonsense Reasoning. *arXiv preprint arXiv:2106.12066*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Vig, J.; and Belinkov, Y. 2019. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284*.

Wang, L.; Li, X.; Liu, J.; He, K.; Yan, Y.; and Xu, W. 2021. Bridge to Target Domain by Prototypical Contrastive Learning and Label Confusion: Re-explore Zero-Shot Learning for Slot Filling. *arXiv preprint arXiv:2110.03572*.

Wang, Y.; Shen, Y.; and Jin, H. 2018. A bi-model based rnn semantic frame parsing model for intent detection and slot filling. *arXiv preprint arXiv:1812.10235*.

Wu, D.; Ding, L.; Lu, F.; and Xie, J. 2020. Slotrefine: A fast non-autoregressive model for joint intent detection and slot filling. *arXiv preprint arXiv:2010.02693*.

Yu, M.; Liu, J.; Chen, Y.; Xu, J.; and Zhang, Y. 2021. Cross-Domain Slot Filling as Machine Reading Comprehension. In *IJCAI*, 3992–3998.